Android Uygulamalarında Güvenlik Zafiyetlerinin Tespiti (Statik Analiz ile) için 2025 Yılında En Son ve En Etkili Teknikler ve Trendler

I. Yönetici Özeti

Mobil cihazlar, günlük yaşamın vazgeçilmez bir parçası haline gelmiş, akıllı telefonlar ve tabletler bilgisayarların işlevselliğini üstlenmiştir.[1] Özellikle açık kaynaklı yapısı nedeniyle Android tabanlı akıllı telefonlar, kötü amaçlı yazılımlar için cazip bir hedef haline gelmiştir.[1] 2025'in ilk çeyreğinde Android kötü amaçlı yazılım örneklerinde %27'lik bir artış ve engellenen tehditlerde %36'lık bir sıçrama yaşanması [2], bu tehditlerin ivme kazandığını ve sürekli geliştiğini göstermektedir. Bu durum, Android uygulamalarının iç yapısını, işleyişini ve temel özelliklerini kapsamlı bir şekilde ortaya koymayı hedefleyen detaylı bir analiz projesi için proaktif güvenlik önlemlerinin ne kadar kritik olduğunu açıkça ortaya koymaktadır.

Bu rapor, Android uygulamalarında güvenlik zafiyetlerinin statik analiz yoluyla tespitine yönelik 2025 yılı ve sonrası için öngörülen en son ve en etkili teknikleri ve trendleri derinlemesine incelemektedir. Yapay zeka (YZ) ve makine öğrenimi (ML) entegrasyonu, gelişmiş program analizi metodolojileri ve yazılım tedarik zinciri bütünlüğüne odaklanma gibi alanlardaki ilerlemeler, statik analizin geleceğini şekillendirmektedir. Bu gelişmeler, artan tehdit ortamına karşı daha akıllı, otomatik ve kapsamlı zafiyet tespit yetenekleri sunarak, mobil uygulama güvenliği alanında paradigma değişimini temsil etmektedir.

II. Giriş: Android Güvenliğinin ve Statik Analizin Gelişen Manzarası

Android Uygulama Güvenliğinin Kritikliği

Günümüzde akıllı telefonlar, tabletler ve kişisel dijital asistanlar gibi mobil cihazlar günlük yaşantımızın vazgeçilmez bir parçasıdır.[1] Yüksek seviyeli bir mobil cihaz, bilgisayarların işlevselliğinin aynısını yerine getirebilmektedir.[1] Android tabanlı akıllı telefonlar, açık kaynaklı bir işletim sistemi olmaları nedeniyle daha savunmasız hale gelmiştir.[1] Bu durum, sistem arızalarına, bellek kaynaklarının israfına, verilerin bozulmasına, kişisel bilgilerin çalınmasına ve bakım maliyetlerinin artmasına neden olabilmektedir.[1] Mobil cihazların finansal işlemler gibi hassas aktiviteler için giderek daha fazla kullanılması [2], güvenliklerini son derece önemli kılmaktadır.

Kötü amaçlı yazılımlar, Android mobil cihazlarını en sık hedef alan tehditler arasındadır.[3] 2019'un ilk yarısında 1,85 milyon olan Android kötü amaçlı yazılım saldırısı sayısı, yıl sonunda 4,18 milyona yükselmiştir.[3] Bu saldırılar giderek daha sinsi hale gelmekte ve mobil sisteme gizlice sızmaktadır.[3] Kullanıcılar, uygulamaları yüklerken izin istekleri konusunda bilgilendirilseler de, riskleri tam olarak kavramadan izinleri verdiklerinde kendilerini güvensiz koşullara ve kötü amaçlı saldırılara maruz bırakmaktadırlar.[3] Android'in açık kaynaklı yapısı ve kullanıcıların izinleri tam olarak anlamadan onaylaması, önemli bir saldırı yüzeyi oluşturmaktadır. Bu yapısal özelliklerin ve kullanıcı davranışlarının birleşimi, statik analizin kod düzeyinde bu tür zafiyetleri tespit etme ve hafifletme yeteneğini temel bir savunma mekanizması haline getirmektedir. Truva atları ve önceden yüklenmiş kötü amaçlı yazılımların yükselişi [2], tespiti daha da karmaşık hale getirmektedir.

Statik Analizin Temel Rolü

Mobil telefon güvenliği, mobil bilişimde son derece önemli bir konudur.[1] Kötü amaçlı mobil uygulamaların tespitindeki zorluklar, önemli sayıda Android kötü amaçlı yazılımın keşfedilmeden aylarca kalmasına neden olmuştur.[1] Bu bağlamda, statik analiz, uygulamayı çalıştırmadan kodunu titizlikle inceleyerek zafiyetleri dağıtımdan önce tespit eden kritik bir önleyici yaklaşımdır.[1] Bu yöntem, çalışma zamanında ortaya çıkmayabilecek veya belirli tetikleyiciler gerektirebilecek sorunları belirleyerek dinamik analizi tamamlamaktadır.[3] Statik analiz, yazılımın semantiğine dayalı bir güvenlik

politikası belirleyerek çalışma zamanı koruması sağlamak veya programdaki hataları saldırıya uğramadan önce tespit etmek için kullanılabilir.[4] Temel amaç, potansiyel kötü amaçlı faaliyetleri sürekli olarak izlemek, tanımlamak ve başlatılmadan önce durdurarak potansiyel hasarı veya kayıpları en aza indirmektir.[1]

III. 2025 Yılında Android Statik Analizinde En Son 10 İleri Teknik ve Trend

Statik analizin evrimi, Android uygulamalarının ve kötü amaçlı yazılımların artan karmaşıklığı nedeniyle temel desen eşleştirmeden sofistike, bağlama duyarlı ve diller arası analize doğru belirgin bir kayma göstermektedir. Yapay zeka ve makine öğrenimi, statik analizi sadece tespit odaklı bir araç olmaktan çıkarıp, yanlış pozitifler ve manuel çaba gibi uzun süredir devam eden sorunları ele alarak düzeltme ve önceliklendirmeye aktif olarak yardımcı olan bir araca dönüştürmektedir. Ayrıca, yazılım tedarik zinciri güvenliğine artan vurgu, statik analizin kapsamını uygulamanın doğrudan kod tabanının ötesine, üçüncü taraf bileşenler ve derleme ortamları dahil olmak üzere tüm bağımlılık ekosistemine genişletmektedir.

1. AI/ML Destekli İzin Tabanlı Kötü Amaçlı Yazılım Tespiti

- **Tanım:** Bu teknik, Android uygulamasının AndroidManifest.xml dosyasından istenen izinleri analiz etmek için makine öğrenimi algoritmalarını, özellikle Random Forest'ı kullanır.[3] Sistem, büyük miktarda iyi niyetli ve kötü amaçlı uygulama veri kümesi üzerinde eğitilerek, kötü amaçlı niyeti gösteren izin isteklerindeki kalıpları öğrenir ve yüksek doğruluk oranlarına ulaşır (örneğin, Random Forest ile %91,6 TPR).[3] Süreç, izinlerin çıkarılması, özellik seçimi (örneğin, Particle Swarm Optimization, Information Gain) ve sınıflandırmayı içerir.[3]
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu yaklaşım, verimliliği ve davranışsal göstergelere dayalı olarak bilinen ve yeni varyantları tespit etme yeteneği nedeniyle ilk kötü amaçlı yazılım taraması için temel ve son derece etkili bir yöntem olmaya devam edecektir.[3] Otomatik uygulama mağazası denetimi, erken tespit için sürekli entegrasyon/sürekli dağıtım (CI/CD) boru hatları ve büyük ölçekli tehdit istihbarat platformları için özellikle faydalıdır.

- **Güvenilir Kaynak/Referans:** Mohamad Arif J. ve diğerleri (2021) "A static analysis approach for Android permission-based malware detection systems." PLoS ONE 16(9): e0257968. [3]

2. Diller Arası Kötü Amaçlı Yazılım Tespiti için Grafik Sinir Ağları (GNN'ler)

- **Tanım:** GNNDroid (2025'te yayınlandı) gibi araçlarda kullanılan GNN'ler, Android uygulamalarını hem Java hem de yerel koddaki kapsamlı davranışları yakalayan Çok İlişkili Yönlü Grafikler (MRDG'ler) olarak temsil eder.[5] Bu, kötü amaçlı yazılımların geleneksel Java merkezli analizi atlatmak için kötü amaçlı davranışları yerel kodda gizleme eğiliminde olmasıyla ortaya çıkan büyüyen zorluğun üstesinden gelir.[5] GNN, kötü amaçlı uygulamaları ayırt etmek için bu grafikleri analiz eder ve yüksek F1 puanları elde eder (örneğin, Java+yerel uygulamalar için %98,57).[5]
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** GNN'ler, daha basit imza veya izin tabanlı yöntemlerden kaçan sofistike, çok dilli kötü amaçlı yazılımları tespit etmek için kritik olacaktır. Karmaşık bileşenler arası etkileşimleri ve veri akışlarını modelleme yetenekleri, özellikle yerel kod kullanımının artmasıyla [6] gelişmiş kötü amaçlı yazılım analizi, tersine mühendislik ve derinlemesine güvenlik denetimleri için vazgeçilmez hale gelecektir.
- **Güvenilir Kaynak/Referans:** GNNDroid: Graph-Learning Based Malware Detection for Android Apps With Native Code. 2025, pp. 1460-1476, vol. 22. IEEE Transactions on Dependable and Secure Computing. [5]

3. Yapay Zeka Destekli Otomatik Zafiyet Giderme (Autofix)

- **Tanım:** Gelişmekte olan Statik Uygulama Güvenlik Testi (SAST) araçları, yalnızca zafiyetleri tespit etmekle kalmayıp aynı zamanda güvenli kod yamaları önermek ve hatta önizlemek için yapay zekayı entegre etmektedir.[7] Aikido Security ve Mend.io gibi araçlarda 2025 için görülen bu "Autofix" yeteneği, geliştiricilerin tespit edilen güvenlik kusurlarını gidermek için gereken manuel çabayı ve süreyi önemli ölçüde azaltmayı amaçlamaktadır.[7] Snyk Code'un DeepCode AI'ı, doğrudan entegre geliştirme ortamında (IDE) hızlı düzeltmeler üretir.[7]
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu trend, güvenlik tespiti ve

düzeltme arasındaki geri bildirim döngüsünü hızlandırarak DevSecOps'ta devrim yaratacaktır. Daha hızlı yama döngülerine, azalan teknik borca ve iyileştirilmiş genel kod güvenlik duruşuna yol açacak, güvenliği çevik geliştirme ortamlarında daha entegre ve daha az bir darboğaz haline getirecektir.
- **Güvenilir Kaynak/Referans:** TheCTOClub.com, "Best Code Analysis Tools" (Makale Son Güncelleme Tarihi: 4 Haziran 2025). [7]

4. Yapay Zeka Ajanları ile Bağlama Duyarlı Statik Analiz

- **Tanım:** Gelişmiş statik analiz, yapay zeka ajanlarının her güvenlik uyarısının bağlamını analiz ettiği, ilgili bilgileri topladığı ve kanıtlarla birlikte bir karar verdiği "bağlama duyarlı" yaklaşımlara doğru ilerlemektedir.[1] Bu, kodun gizlenmesini çözmek için komut dosyaları oluşturma ve yürütme yeteneğini içerir ve "tam görünürlükle daha hızlı tespit ve yanıt" sağlar.[8] Bu, daha fazla eşdeğerlik sınıfını ayırt edebilen daha hassas bir statik analiz sürümüdür.[4]
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu teknik, uyarıların önceliklendirilmesini ve incelenmesini otomatikleştirerek güvenlik analistlerinin manuel iş yükünü büyük ölçüde azaltacaktır.[8] Belirli program durumlarına veya veri akışlarına bağlı olan ince zafiyetleri ve gizlenmiş kötü amaçlı kodu analiz etmek için kritik öneme sahiptir. Uygulaması sürekli izleme ve gerçek zamanlı tehdit istihbaratına kadar uzanmaktadır.
- **Güvenilir Kaynak/Referans:** Google Cloud Blog, "Next '25: Driving secure innovation with AI, Google Unified Security" (Önizleme Q2 2025 bekleniyor). [1]

5. Yerel Kod ve Diller Arası Akışlar için Taint Analizi

- **Tanım:** Geleneksel taint analizi, Android uygulamalarında, kötü amaçlı olanlar da dahil olmak üzere giderek daha fazla kullanılan yerel kodun veri akışı davranışlarıyla genellikle zorlanmaktadır.[6] JNFuzz-Droid (Mayıs 2025'te yayınlandı) gibi yeni çerçeveler, Android yerel kodundaki hassas veri sızıntısını veya transferini etkili bir şekilde tespit etmek için fuzzing'i taint analizi ile birleştirir ve önceki en son teknoloji araçlardan daha iyi performans gösterir.[6] Bu, diller arası iletişimi ve Java Native Interface (JNI) katmanındaki potansiyel zafiyetleri anlamak için kritik öneme sahiptir.

- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu teknik, Android güvenlik analizindeki kritik bir kör noktayı doğrudan ele almaktadır. Kötü amaçlı yüklerini veya veri sızdırma rutinlerini yerel kütüphanelerde gizleyen sofistike kötü amaçlı yazılımları tespit etmek ve karmaşık uygulamalarda Java ve yerel bileşenler arasında hassas verilerin güvenli bir şekilde işlenmesini sağlamak için vazgeçilmez olacaktır.
- **Güvenilir Kaynak/Referans:** Jianchao Cao ve diğerleri (2025) "JNFuzz-Droid: a lightweight fuzzing and taint analysis framework for native code of Android applications." Empirical Software Engineering 30(5). [6]

6. İkili Düzey Güvenlik ve Etki Karakterizasyonu için Sembolik Yürütme

- **Tanım:** Hata avcılığı ve otomatik test için güçlü bir teknik olan sembolik yürütme, ikili düzey güvenlik incelemeleri için uyarlanmaktadır.[9] Bu, üçüncü taraf bileşenler üzerindeki zafiyet analizi, yan kanal saldırıları ve kötü amaçlı yazılım tersine mühendisliğini içerir.[9] Sid gibi araçlar, kuralları karşılaştırmak ve uygulanmamış yamaların güvenlik etkilerini belirlemek için sembolik yürütmeyi kullanır ve yüksek hassasiyetle (örneğin, %97) zafiyetleri sınıflandırır.[10]
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu teknik, özellikle kapalı kaynak ikililer, aygıt yazılımı ve kritik sistem bileşenleri için derinlemesine güvenlik değerlendirmeleri için hayati önem taşıyacaktır.[9] Zafiyetlerin ve yamaların etkisini hassas bir şekilde karakterize etme yeteneği, Android çekirdeği gibi kritik bileşenlerdeki risk yönetimi ve önceliklendirme çabalarını önemli ölçüde artıracaktır.[10]
- **Güvenilir Kaynak/Referans:** PLDI 2025 Çalıştayı: "BINSEC: Adapting Symbolic Execution for Binary-level Security.".[9] NDSS Sempozyumu: "Precisely Characterizing Security Impact in a Flood of Patches via Symbolic Rule Comparison." [10]

7. Hedefli Statik Analiz için Tahminsel Analitik

- **Tanım:** Yapay zeka/makine öğrenimi destekli tahminsel modeller, geçmiş zafiyet verilerini, kod değişikliği günlüklerini ve kusur eğilimlerini analiz ederek bir uygulamanın hangi alanlarının yeni veya keşfedilmemiş güvenlik kusurları içerme

olasılığının en yüksek olduğunu belirler.[12] Bu, statik analiz araçlarının taramalarını yüksek riskli modüllere veya kod bölümlerine önceliklendirmesini sağlayarak analiz sürecini daha verimli ve odaklı hale getirir.
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu teknik, statik analizi geniş, genellikle zaman alıcı bir taramadan akıllı, risk odaklı bir yaklaşıma dönüştürür. Büyük kod tabanları, sürekli entegrasyon ortamları ve güvenlik ekiplerinin kaynakları etkili bir şekilde tahsis etmesi için paha biçilmez olacak, en kritik alanların titizlikle incelenmesini sağlayacaktır.
- **Güvenilir Kaynak/Referans:** ResearchGate.net, "Future of Software Test Automation Using AI/ML" (Makale Son Güncelleme Tarihi: 16 Mayıs 2025). [12]

8. Sıfır Gün Zafiyetleri için Kod Desenlerinde Anomali Tespiti

- **Tanım:** Denetimsiz öğrenme kullanılarak, statik analiz araçları alışılmadık kod desenlerini veya yerleşik güvenli kodlama standartlarından sapmaları tespit edebilir.[12] Kod tabanını analiz ederek, bu sistemler, önceden tanımlanmış kurallarla kapsanmayan yeni saldırı vektörlerini veya sıfır gün zafiyetlerini (örneğin, beklenmedik fonksiyon çağrı dizileri veya veri akışı düzensizlikleri) gösterebilecek anormallikleri belirler.
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu teknik, daha önce bilinmeyen zayıflıkları kullanan ortaya çıkan tehditlere karşı proaktif bir savunma sunar. Özellikle sofistike, sinsi kötü amaçlı yazılımları veya mevcut imzalarla eşleşmeyen yeni istismar tekniklerini tespit etmek için faydalıdır, böylece Android uygulamalarının gelişen tehditlere karşı direncini artırır.
- **Güvenilir Kaynak/Referans:** ResearchGate.net, "Future of Software Test Automation Using AI/ML" (Makale Son Güncelleme Tarihi: 16 Mayıs 2025). [12]

9. Statik Analizin Yazılım Tedarik Zinciri Güvenliği ile Entegrasyonu

- **Tanım:** 2025'te yazılım tedarik zinciri saldırılarının hızlanması ve evrimi ile [13], statik analiz, üçüncü taraf kütüphaneler ve açık kaynak bağımlılıkları dahil olmak üzere tüm bileşenlerin bütünlüğünü ve güvenliğini doğrulamak için ayrılmaz hale gelmektedir.[13] Bu, hem tescilli hem de açık kaynak kodundaki sızdırılmış geliştirici sırlarını, güvensiz tasarımı ve yetersiz uygulama sertleştirmesini tespit etmeyi

içerir.[13] Odak noktası, sadece CVE'lerin ötesine geçerek kurcalama ve sırların maruz kalması gibi daha geniş risklere kaymaktadır.[13]
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Statik analiz, yazılım tedarik zincirinde güven ve şeffaflık oluşturmak için bir köşe taşı olacaktır. Kuruluşların, Android uygulamalarının harici bağımlılıklar aracılığıyla ortaya çıkan zafiyetler tarafından tehlikeye atılmamasını sağlayarak bileşenleri entegrasyondan önce denetlemesini sağlayacaktır. Bu, önceden yüklenmiş kötü amaçlı yazılımlardan ve kötü amaçlı paketlerden kaynaklanan riskleri azaltmak için kritik öneme sahiptir.[2]
- **Güvenilir Kaynak/Referans:** ISACA, "The 2025 Software Supply Chain Security Report: Threats Growing and Evolving" (Makale Son Güncelleme Tarihi: 5 Mayıs 2025). [13]

10. OWASP MASVS Uyumlu Statik Analiz

- **Tanım:** Statik analiz araçları, OWASP Mobil Uygulama Güvenliği Doğrulama Standardı (MASVS) gibi endüstri standardı güvenlik çerçeveleriyle giderek daha fazla uyum sağlamaktadır.[15] APKHunt (OWASP MASVS v1.5.0, Ocak 2023'e dayalı) gibi araçlar, çeşitli MASVS gereksinimleri genelinde tarama kapsamı sunar ve düşük yanlış pozitifler ve belirli güvenlik sinkleri için optimize edilmiş tarama üzerine odaklanır.[15]
- **2025 Potansiyel Etkileri ve Uygulama Alanları:** Bu trend, standartlaştırılmış güvenlik değerlendirmelerini kolaylaştırır ve geliştiricilerin ve güvenlik test uzmanlarının sonuçlarının eksiksizliğini ve tutarlılığını sağlamasına yardımcı olur.[15] Belirli güvenlik uyumluluk seviyelerine ulaşmak, kapsamlı kod incelemeleri yapmak ve geniş bir mobil uygulama güvenlik riski yelpazesini sistematik olarak ele almak isteyen kuruluşlar için gereklidir.
- **Güvenilir Kaynak/Referans:** GitHub - Cyber-Buddy/APKHunt (OWASP MASVS v1.5.0'a dayalı, Ocak 2023'te yayınlandı). [15]

IV. Yapay Zeka ve Makine Öğreniminin Statik Analizdeki Dönüştürücü Rolü

Yapay zeka (YZ) ve makine öğrenimi (ML) teknolojileri, yazılım test otomasyonunda

önemli bir boşluğu doldurarak, minimum insan müdahalesiyle uyum sağlayabilen, öğrenebilen ve tahminsel kararlar alabilen mekanizmalar sunmaktadır.[12] Bu teknolojiler, statik analizin yeteneklerini önemli ölçüde genişletmekte ve güvenlik analizini daha verimli ve etkili hale getirmektedir.

Tespit ve Verimliliği Artırma

YZ/ML'nin statik analizdeki en doğrudan uygulamalarından biri, kötü amaçlı yazılım tespitinin doğruluğunu ve verimliliğini artırmaktır. Android uygulamalarını görüntü olarak ele alma ve her uygulama için komşuluk matrisleri oluşturma gibi derin öğrenme yaklaşımları (AdMat), farklı kötü amaçlı yazılım veri kümelerinde %98,26 gibi etkileyici ortalama tespit oranları elde etmiştir.[1] Bu, YZ'nin karmaşık kötü amaçlı davranış kalıplarını geleneksel yöntemlerden daha etkili bir şekilde tanımlayabildiğini göstermektedir.

Makine öğrenimi sınıflandırıcıları, Android kötü amaçlı yazılım tespiti için yaygın olarak kullanılmaktadır. Özellikle izin tabanlı özelliklerle birlikte Naive Bayes, destek vektör makineleri (SVM), karar ağaçları ve Random Forest gibi modeller, kötü amaçlı yazılım tespitinde başarılı sonuçlar vermiştir.[3] Random Forest algoritması, kötü amaçlı yazılım tespitinde %91,6'lık bir Doğru Pozitif Oranı (TPR) ile en yüksek doğruluk seviyesine ulaşmıştır.[3] Bu algoritmalar, büyük veri kümelerinden öğrenerek, bir uygulamanın izin isteklerine veya API çağrılarına dayanarak kötü amaçlı olup olmadığını belirleyebilir.[3]

Tespitin Ötesinde: Otomasyon ve Önceliklendirme

YZ/ML'nin statik analize uygulanması, sadece tespit doğruluğunu artırmakla kalmayıp, güvenlik yaşam döngüsünün tamamında akıllı otomasyonu mümkün kılmaktadır. Bu, gereksinim analizinden otomatik düzeltmeye kadar uzanan bir dönüşümü ifade eder.

- **Doğal Dil İşleme (NLP):** NLP, makinelerin insan dilini yorumlamasına ve işlemesine olanak tanır.[12] Güvenlik bağlamında, NLP, güvenlik gereksinim belgelerini, kullanıcı hikayelerini veya hata raporlarını otomatik olarak yapılandırılmış test senaryolarına dönüştürerek manuel betik oluşturma yükünü azaltabilir.[12] Bu, güvenlik politikalarının kod düzeyinde otomatik olarak uygulanmasını ve doğrulanmasını sağlayabilir.

- **Tahminsel Analitik:** Tahminsel modeller, geçmiş zafiyet verilerini, kod değişikliği günlüklerini ve kusur eğilimlerini analiz ederek uygulamanın hangi alanlarının yeni veya keşfedilmemiş güvenlik kusurları içerme olasılığının en yüksek olduğunu belirler.[12] Bu yetenek, statik analiz araçlarının taramalarını yüksek riskli modüllere veya kod bölümlerine odaklamasına olanak tanır. Bu, analiz sürecini daha verimli ve odaklı hale getirir, zaman kazandırır ve erken kusur tespit olasılığını artırır.[12]
- **Anomali Tespiti:** YZ/ML sistemleri, denetimsiz öğrenme tekniklerini (kümeleme, PCA ve otoenkoderler gibi) kullanarak alışılmadık kod desenlerini veya yerleşik güvenli kodlama standartlarından sapmaları tespit edebilir.[12] Bu, önceden tanımlanmış kurallarla kapsanmayan yeni saldırı vektörlerini veya sıfır gün zafiyetlerini (örneğin, beklenmedik fonksiyon çağrı dizileri veya veri akışı düzensizlikleri) işaretleyebilir. Bu, gelişen tehditlere karşı proaktif bir savunma sunar.
- **Pekiştirmeli Öğrenme:** Pekiştirmeli öğrenme algoritmaları, sınırlı zaman veya hesaplama bütçeleri dahilinde test kapsamını maksimize etmek ve hataları bulmak için gerçek zamanlı kararlar alabilir.[12] Bu ajanlar, ortamla etkileşime girerek en iyi yürütme yollarını öğrenir ve bir zafiyet bulma gibi "ödüllere" dayalı olarak analiz stratejilerini dinamik olarak ayarlayabilirler.

SAST Araç Sınırlamalarını Giderme

YZ destekli özellikler, SAST araçlarının uzun süredir devam eden sorunlarını ele almak için özel olarak tasarlanmıştır:

- **YZ Destekli Otomatik Düzeltme/Giderme:** Aikido Security ve Mend.io gibi platformlar, YZ destekli "Autofix" önizlemeleri sunarak güvenli kod yamaları önerir ve hatta tek tıklamayla uygulanabilecek düzeltmeler sağlar.[7] Bu, çözüme ulaşma süresini önemli ölçüde kısaltır ve geliştiricilerin manuel çabasını azaltır. Snyk Code'un DeepCode AI aracı, sorunları tanımlarken hızlı düzeltmeler listesi oluşturur ve bu düzeltmeler doğrudan IDE'den incelenebilir ve uygulanabilir.[7]
- **Yanlış Pozitifleri Azaltma:** DerScanner'ın Confi AI motoru gibi YZ destekli motorlar, yanlış pozitifleri en aza indirmek için tasarlanmıştır.[7] Bu, güvenlik ekiplerinin gereksiz uyarılar yerine gerçek güvenlik risklerine odaklanmasını sağlar ve uyarı yorgunluğunu azaltır.

Zorluklar ve Değerlendirmeler

YZ/ML'nin statik analize entegrasyonu önemli faydalar sunsa da, bazı zorluklar devam etmektedir:

- **Veri Kullanılabilirliği:** Doğru ML modellerini eğitmek için büyük, yapılandırılmış veri kümeleri gereklidir, ancak birçok kuruluş bu tür verileri düzenli formatlarda tutmamaktadır.[12]
- **Model Yorumlanabilirliği:** "Kara kutu" YZ modelleri, test sonuçlarını veya hataları anlamayı zorlaştıran düşük şeffaflık sunabilir, bu da güveni ve hata ayıklamayı etkiler.[12]
- **Araç Uyumluluğu:** Birçok eski sistem, modern YZ/ML test platformlarıyla entegrasyon için gerekli altyapıya sahip değildir.[12]
- **Yetenek Açığı:** Hem yazılım testini/güvenliğini hem de makine öğrenimini anlayan yetenekli profesyonellerde önemli bir açık bulunmaktadır.[12]
- **Maliyet ve Karmaşıklık:** Gelişmiş YZ/ML çözümlerinin uygulanması, altyapı, araçlar ve eğitim için başlangıç yatırımı gerektirebilir.[12]
- **Cihaz Üzeri YZ'de Performans ve Pil Optimizasyonu:** Mobil uygulama geliştirmede "Cihaz Üzeri Zeka (Edge AI ve Gizlilik Odaklı Tasarım)" [17] trendi, statik analizin yeni bir sınırla karşı karşıya olduğunu göstermektedir. Hassas veri işleme ve yapay zeka çıkarımı cihaza kaydıkça, statik analizin bu cihaz üzeri yapay zeka bileşenlerinin güvenli uygulamasını, veri işlemeyi ve kurcalanmaya karşı direncini doğrulamak için evrilmesi gerekecektir. Bu, performans ve pil optimizasyonu arasında denge kurma zorluğunu da beraberinde getirmektedir.[17]

Tablo 1: Yapay Zeka/Makine Öğrenimi Destekli Statik Analiz Özellikleri (2025)

Özellik Adı	Nasıl Çalışır	Güvenlik/Gelişime Birincil Faydası	Örnek Araç	İlgili Kaynak ID(leri)
Yapay Zeka Destekli Otomatik Düzeltme/Giderme	Zafiyetleri tespit eder ve güvenli kod yamaları önerir/önizler.	Daha hızlı düzeltme, manuel çabayı azaltma.	Aikido Security, Mend.io, Snyk Code	[7]
Akıllı Yanlış	Yapay zeka	Uyarı	DerScanner	[7]

Pozitif Azaltma	motorları, yanlış pozitifleri en aza indirerek gerçek risklere odaklanmayı sağlar.	yorgunluğunu azaltma, analiz doğruluğunu artırma.		
Tahminsel Analitik	Geçmiş verileri analiz ederek zafiyet olasılığı yüksek alanları belirler ve taramaları önceliklendirir.	Hedefli taramalar, erken kusur tespiti, zaman tasarrufu.	-	12
Anomali Tespiti	Denetimsiz öğrenme ile alışılmadık kod desenlerini veya güvenli kodlama standartlarından sapmaları belirler.	Sıfır gün zafiyetlerinin tespiti, yeni saldırı vektörlerine karşı koruma.	-	12
Bağlama Duyarlı Analiz (AI Ajanları ile)	Her güvenlik uyarısının bağlamını analiz eder, ilgili bilgileri toplar ve karar verir; kodun gizlenmesini çözme yeteneği.	Analist iş yükünü azaltma, daha hızlı ve kapsamlı tespit.	Google Threat Intelligence	1
Diller Arası Kötü Amaçlı Yazılım Tespiti (GNN'ler)	Uygulamaları çok ilişkili grafikler olarak temsil eder ve Java ile yerel kod arasındaki kötü amaçlı davranışları analiz eder.	Karmaşık, çok dilli kötü amaçlı yazılımların tespiti.	GNNDroid	5
Yerel Kod Taint Analizi	Hassas veri akışını yerel kodda izler,	Yerel koddaki gizli zafiyetlerin tespiti, veri	JNFuzz-Droid	6

	sızıntı veya transferi tespit eder.	güvenliği.		
İkili Düzey Sembolik Yürütme	Program yollarını sembolik olarak keşfeder, ikili düzeyde zafiyet analizi ve etki karakterizasyon u yapar.	Kapalı kaynak bileşenlerin derinlemesine analizi, yama etkisinin belirlenmesi.	BINSEC, Sid	[9]

V. Gelişmiş Program Analiz Metodolojileri

Android kötü amaçlı yazılımlarının artan karmaşıklığı, özellikle yerel kod kullanımı ve gizleme teknikleri, daha derinlemesine, daha hassas ve diller arası program analiz metodolojilerine olan ihtiyacı artırmaktadır. Bu gelişmiş yaklaşımlar, basit zafiyet sayımının ötesine geçerek güvenlik politikaları, veri akışları ve zafiyetlerin etkisi hakkında daha derin bir anlayış sağlayarak daha stratejik ve etkili düzeltmelere olanak tanır.

Bağlama Duyarlı Analiz

Statik program analizi, programı çalıştırmadan program hakkında bilgi hesaplar.[4] Bağlama duyarlı analiz, bu bilginin hesaplanmasında belirli yürütme bağlamını (örneğin, fonksiyon çağrı siteleri, nesne durumları) dikkate alan gelişmiş bir biçimdir. Bu, veri akışları ve değişken durumları hakkında bağlama duyarsız analizden daha hassas içgörüler sağlar.[4] Güvenlik uygulamalarında, bu, program değişkenlerini daha hassas bir şekilde sınıflandırmak ve bunları benzersiz anahtarlarla şifrelemek için Veri Alanı Rastgeleleştirmesi (DSR) için kritik öneme sahiptir, böylece bellek bozulması saldırılarını engeller.[4] Ayrıca, çekirdek ASLR atlamalarına yol açabilecek doğrudan adres ifşalarının tespitini de sağlar.[4]

Gelişmiş Taint Analizi

Taint analizi, hassas verilerin (taint kaynakları) bir uygulama aracılığıyla akışını izleyerek nerede ifşa edilebileceğini veya kötüye kullanılabileceğini (taint sinkleri) belirler.[6] Geleneksel taint analizi, Android uygulamalarında, kötü amaçlı olanlar da dahil olmak üzere giderek daha fazla kullanılan yerel kodun veri akışı davranışlarıyla genellikle zorlanmaktadır.[6] JNFuzz-Droid (Mayıs 2025'te yayınlandı) gibi yeni çerçeveler, fuzzing'i taint analizi ile birleştirerek Android'in yerel kodundaki hassas veri sızıntısını veya transferini etkili bir şekilde tespit etme zorluğunu doğrudan ele almaktadır.[6] Bu, kötü amaçlı uygulamaların işlevselliği yerel kütüphanelerde gizlemesi nedeniyle kritik öneme sahiptir.[5] JNFuzz-Droid'in deneysel sonuçları, yerel kodda hassas veri sızıntısını veya transferini etkili bir şekilde tespit edebildiğini ve en son teknoloji yerel analiz araçlarından daha iyi performans gösterdiğini göstermektedir.[6]

Sembolik Yürütme

Sembolik yürütme, girişleri somut değerler yerine sembolik değişkenler olarak ele alarak program yollarını keşfeden, yol koşulları üreten ve zafiyetleri belirleyen güçlü bir tekniktir.[9] Önemli gelişmeler, sembolik yürütmenin ikili düzey güvenlik incelemelerine uyarlanmasında kaydedilmektedir. Bu, üçüncü taraf bileşenler üzerindeki zafiyet analizi, yan kanal saldırıları ve kötü amaçlı yazılım tersine mühendisliğini içerir.[9] Örneğin, Sid gibi araçlar, uygulanmamış bir yamanın güvenlik etkilerini belirlemek için kural karşılaştırması ve eksik kısıtlı sembolik yürütme kullanır. Sid, zafiyetleri güvenlik etkilerine göre otomatik olarak sınıflandırabilir ve 66 bin son taahhütten 227 güvenlik hatası tespit etmiştir; bunların 197'si daha önce bildirilmemiştir.[10] Bu, sembolik yürütmenin sadece hata bulmanın ötesine geçerek, zafiyetlerin sonuçlarını ve daha geniş güvenlik hedefleriyle (örneğin, veri bütünlüğü, gizlilik, sistem kontrolü) nasıl ilişkili olduğunu anlamaya yardımcı olduğunu göstermektedir. Bu, statik analiz yeteneklerinde niteliksel bir sıçramayı temsil eder.

VI. Statik Analiz Yoluyla Android Yazılım Tedarik Zincirini Güvenli

Hale Getirme

Yazılım tedarik zinciri güvenliğine artan vurgu, statik analizin Android uygulamaları için kapsamını ve kritikliğini temelden yeniden tanımlamaktadır. Bu, odak noktasının bireysel uygulama zafiyetlerinden, tüm yazılım tedarik zincirinin bütünsel güvenliğine kaydığını göstermektedir.

Yükselen Tehdit Ortamı

Yazılım tedarik zinciri saldırıları, 2025 yılında hızlanmakta ve hızla evrim geçirmektedir.[13] 2020'deki SolarWinds saldırısı bu saldırı vektörünü ön plana çıkarmış ve tehditler hem açık kaynaklı (OSS) hem de kapalı kaynaklı ticari yazılımlarda artmaya devam etmektedir.[13] Temel riskler arasında açık kaynaklı depolardaki kötü amaçlı paketler, sızdırılmış geliştirici sırları (sabit kodlu kimlik bilgileri, API ve şifreleme anahtarları), güvensiz tasarım ve yetersiz uygulama sertleştirmesi bulunmaktadır.[13] Triada arka kapısı gibi kötü amaçlı yazılımlar, geleneksel uygulama mağazası denetimini atlayarak cihazlara önceden yüklenebilir veya uygulamalara pazara ulaşmadan önce enjekte edilebilir.[2] Bu, statik analizin sadece uygulamanın kendi kodunu değil, tüm bağımlılıklarını, derleme yapıtlarını ve üretim hattını da incelemesi gerektiğini zorunlu kılmaktadır.

Bir Hafifletme Stratejisi Olarak Statik Analiz

Statik analiz, üçüncü taraf kütüphaneler ve açık kaynak bileşenler aracılığıyla ortaya çıkan zafiyetleri tespit etmek için kritik öneme sahiptir.[14] Kuruluşlar, yalnızca iyi denetlenmiş, sık güncellenen kütüphaneleri kullanmalı ve güvenilmeyen kaynaklardan kaçınmalıdır.[14] Odak noktası, geleneksel Ortak Zafiyetler ve Açıklamalar (CVE) tabanlı zafiyet yönetiminden, sızdırılmış sırlar, derleme ortamının kurcalanması ve dosya bozulması gibi daha geniş yazılım tedarik zinciri risklerine kaymaktadır.[13] ABD Ulusal Standartlar ve Teknoloji Enstitüsü'nün (NIST) 2024'te CVE'leri zenginleştirmeyi bırakma kararı, güvenlik liderlerinin yazılım ürünlerindeki güvenlik risklerini nasıl yöneteceklerini yeniden düşünmeleri gerektiğini vurgulamaktadır.[13] Karmaşık ikili analiz ve

tekrarlanabilir derlemeler gibi teknolojiler, gerekli görünürlüğü sağlamak için anahtar hale gelmektedir.[13]

Blok Zincirinin Potansiyeli (Dolaylı Bağlantı)

Blok zinciri teknolojisi, tedarik zinciri şeffaflığını ve direncini artırma potansiyeli sunsa da, birlikte çalışabilirlik ve ölçeklenebilirlik gibi çeşitli zorluklarla karşılaşmaktadır.[18] Özellikle halka açık blok zincirleri, işlem gücü, yüksek hızlı internet bağlantısı, enerji tüketimi ve depolama alanı gibi ölçeklenebilirlik sorunları yaşamaktadır.[18] Bu zorluklar, blok zincirinin tipik Android geliştirme süreçlerinde statik analiz doğrulaması için yaygın entegrasyonunun 2025'e kadar olgunlaşmayabileceği anlamına gelmektedir. Statik analiz bağlamında, blok zinciri potansiyel olarak yazılım bileşenlerini ve güvenlik onaylarını izlemek için değişmez bir defter sağlayabilir ve statik analiz araçları daha sonra bunları doğrulayabilir. Ancak, 2025'te Android güvenliği için statik analize doğrudan uygulaması hala başlangıç aşamasındadır ve önemli engellerle karşılaşmaktadır.[18] Bu nedenle, doğrudan pratik etkisi sınırlı kalacaktır.

VII. Stratejik Değerlendirmeler ve Gelecek Görünümü

Statik analizdeki önemli teknolojik ilerlemelere rağmen, insan faktörü ve organizasyonel zorluklar, 2025'te etkili uygulama ve benimseme için kritik darboğazlar olmaya devam etmektedir. Gelişmiş teknoloji tek başına yeterli değildir; sosyo-teknik faktörler de gerçek dünya etkisi için eşit derecede önemlidir.

Benimsemedeki Zorluklar

- **Veri Kullanılabilirliği:** Doğru yapay zeka/makine öğrenimi modellerini eğitmek için büyük, yapılandırılmış veri kümeleri gereklidir, ancak birçok kuruluş bu tür verileri düzenli formatlarda tutmamaktadır.[12]
- **Model Yorumlanabilirliği:** "Kara kutu" yapay zeka modelleri, test sonuçlarını veya hataları anlamayı zorlaştıran düşük şeffaflık sunabilir, bu da güveni ve hata

ayıklamayı etkiler.[12]
- **Araç Uyumluluğu:** Birçok eski sistem, modern yapay zeka/makine öğrenimi test platformlarıyla entegrasyon için gerekli altyapıya sahip değildir.[12]
- **Yetenek Açığı:** Hem yazılım güvenliği/testi hem de makine öğrenimini anlayan yetenekli profesyonellerde önemli bir açık bulunmaktadır.[12]
- **Maliyet ve Karmaşıklık:** Gelişmiş yapay zeka/makine öğrenimi çözümlerinin uygulanması, altyapı, araçlar ve eğitim için başlangıç yatırımı gerektirebilir.[12]

Fırsatlar ve Öneriler

Android statik analizinin geleceği, izole araçlardan ziyade, kapsamlı bir DevSecOps çerçevesine ve hem teknik zafiyetleri hem de insan faktörlerini ele alan daha geniş, proaktif bir güvenlik duruşuna sorunsuz entegrasyonla ilgilidir.

- **DevSecOps Entegrasyonu:** Güvenliğin geliştirme döngüsünün başından itibaren dahil edilmesi esastır. Otomatik statik ve dinamik analiz araçlarının kullanılması, güvenliğin dağıtım hızını engellemeden başlangıçtan itibaren dahil edilmesini sağlar.[14]
- **Sürekli Güvenlik Testi:** Güvenlik testi, zafiyetleri erken tespit etmek için otomatik araçlardan yararlanılarak sürekli bir süreç olmalıdır.[14]
- **Güvenli Kodlama Uygulamalarına Odaklanma:** Araçların ötesinde, temel güvenli kodlama uygulamaları hala büyük önem taşımaktadır. Bunlar arasında kimlik bilgisi maruziyetini en aza indirme, güvenli kimlik doğrulama (parola anahtarları, biyometrik, MFA) kullanma, verileri hem depoda hem de aktarımda şifreleme ve üçüncü taraf kütüphanelerin dikkatli kullanımı yer alır.[14]
- **Düzenli Güncellemeler:** İşletim sistemi sürümlerini ve üçüncü taraf kütüphanelerini güncel tutmak, bilinen açıklara karşı korunmak için kritik öneme sahiptir.[14]
- **CVE'lerin Ötesi:** Güvenlik liderleri, zafiyet yönetimini yeniden düşünerek, sızdırılmış sırlar ve kurcalama gibi yazılım tedarik zinciri risklerinin tamamına odaklanmalıdır.[13]
- **Kullanıcı Eğitimi:** Sosyal mühendislik ve önceden yüklenmiş kötü amaçlı yazılım tehditlerinin devam etmesi göz önüne alındığında, uygulama izinleri ve güvenli uygulama yükleme uygulamaları (örneğin, resmi uygulama mağazaları, incelemeleri kontrol etme) konusunda kullanıcı dijital okuryazarlığını artırmak kritik öneme sahiptir.[2]

VIII. Sonuç

2025 yılında Android uygulama güvenliği ortamı, artan bir tehdit ortamı ile karakterize edilmektedir ve zafiyet tespitine proaktif ve çok yönlü bir yaklaşım gerektirmektedir. Yapay zeka/makine öğrenimi, gelişmiş program analizi metodolojileri ve yazılım tedarik zincirine genişletilmiş bir odaklanma ile güçlendirilmiş statik analiz, bu savunmanın temel taşı olarak öne çıkmaktadır.

Belirlenen başlıca teknikler, diller arası kötü amaçlı yazılım tespiti için Grafik Sinir Ağlarından, yapay zeka destekli otomatik düzeltmeye ve ikili analiz için sembolik yürütmeye kadar, daha akıllı, hassas ve verimli güvenlik analizine doğru bir paradigma kaymasını temsil etmektedir. Bu ilerlemeler, statik analizi sadece bir tespit aracı olmaktan çıkarıp, tüm güvenlik yaşam döngüsünü kapsayan akıllı bir asistan haline getirmektedir.

Benimsemedeki zorluklar (veri kullanılabilirliği, model yorumlanabilirliği, yetenek açığı gibi) devam etse de, DevSecOps içinde stratejik entegrasyon, sürekli güvenlik uygulamaları ve risklerin bütünsel bir anlayışı, kuruluşların gelişen siber tehditler karşısında daha dirençli Android uygulamaları oluşturmasını sağlayacaktır. Android güvenliğinin geleceği, kötü amaçlı aktörlerin önünde kalmak için bu en son statik analiz yeteneklerinden yararlanmaya bağlıdır.

Alıntılanan çalışmalar

1. Detecting software vulnerabilities in android using static analysis ..., erişim tarihi Haziran 28, 2025, https://www.researchgate.net/publication/286584054_Detecting_software_vulnerabilities_in_android_using_static_analysis
2. Android under fire: Mobile attacks soar in early 2025 - Varindia, erişim tarihi Haziran 28, 2025, https://www.varindia.com/news/android-under-fire-mobile-attacks-soar-in-early-2025
3. A static analysis approach for Android permission-based malware ..., erişim tarihi Haziran 28, 2025, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0257968
4. Security Applications of Static Program Analysis - eScholarship.org, erişim tarihi Haziran 28, 2025, https://escholarship.org/uc/item/4tm5955w
5. GNNDroid: Graph-Learning Based Malware Detection for Android Apps With

Native Code, erişim tarihi Haziran 28, 2025, https://www.computer.org/csdl/journal/tq/2025/02/10638211/1ZsLaQrfqAE
6. JNFuzz-Droid: a lightweight fuzzing and taint analysis framework for native code of Android applications - ResearchGate, erişim tarihi Haziran 28, 2025, https://www.researchgate.net/publication/391952869_JNFuzz-Droid_a_lightweight_fuzzing_and_taint_analysis_framework_for_native_code_of_Android_applications
7. 20 Best Code Analysis Tools in 2025 - The CTO Club, erişim tarihi Haziran 28, 2025, https://thectoclub.com/tools/best-code-analysis-tools/
8. Next '25: Driving secure innovation with AI, Google Unified Security | Google Cloud Blog, erişim tarihi Haziran 28, 2025, https://cloud.google.com/blog/products/identity-security/driving-secure-innovation-with-ai-google-unified-security-next25
9. BINSEC: Adapting Symbolic Execution for Binary-level Security - PLDI 2025, erişim tarihi Haziran 28, 2025, https://pldi25.sigplan.org/details/pldi-2025-PLDI-Workshops/2/BINSEC-Adapting-Symbolic-Execution-for-Binary-level-Security-
10. Precisely Characterizing Security Impact in a Flood of Patches via Symbolic Rule Comparison - NDSS Symposium, erişim tarihi Haziran 28, 2025, https://www.ndss-symposium.org/ndss-paper/precisely-characterizing-security-impact-in-a-flood-of-patches-via-symbolic-rule-comparison/
11. Proceedings of the 18th USENIX WOOT Conference on Offensive Technologies (WOOT '24), erişim tarihi Haziran 28, 2025, https://www.usenix.org/system/files/woot24-proceedings-interior.pdf
12. (PDF) Future of Software Test Automation Using AI/ML - ResearchGate, erişim tarihi Haziran 28, 2025, https://www.researchgate.net/publication/391806293_Future_of_Software_Test_Automation_Using_AIML
13. The 2025 Software Supply Chain Security Report: Threats Growing and Evolving - ISACA, erişim tarihi Haziran 28, 2025, https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2025/the-2025-software-supply-chain-security-report
14. Top 15 Mobile Application Security Best Practices in 2025 - Cybersecurity ASEE, erişim tarihi Haziran 28, 2025, https://cybersecurity.asee.io/blog/top-mobile-application-security-best-practices/
15. Cyber-Buddy/APKHunt: APKHunt is a comprehensive static ... - GitHub, erişim tarihi Haziran 28, 2025, https://github.com/Cyber-Buddy/APKHunt
16. arXiv:2502.15041v1 [cs.CR] 20 Feb 2025, erişim tarihi Haziran 28, 2025, https://arxiv.org/pdf/2502.15041
17. AI Trends in Future of Mobile App Development - Codewave, erişim tarihi Haziran 28, 2025, https://codewave.com/insights/ai-trends-future-mobile-app-development/
18. Using Blockchain to Drive Supply Chain Transparency and Innovation | Deloitte US, erişim tarihi Haziran 28, 2025,

https://www.deloitte.com/us/en/services/consulting/articles/blockchain-supply-chain-innovation.html
19. Security checklist | Android Developers, erişim tarihi Haziran 28, 2025, https://developer.android.com/privacy-and-security/security-tips