Android Platformunda Kötücül Yazılım Tespiti: Literatür İncelemesi

Literatür Makalesi/Review Article

Gökçer PEYNİRCİ[1], Mete EMİNAĞAOĞLU[2]

[1]AttackVector, CyberSecurity Services, Kiev, 01004, Ukrayna
[2]Bilgisayar Bilimleri Bölümü, Dokuz Eylül Üniversitesi, İzmir, Türkiye
gokcerp@gmail.com, mete.eminagaoglu@deu.edu.tr
(Geliş/Received:08.02.2019; Kabul/Accepted:26.01.2020)
DOI: 10.17671/gazibtd.524408

Özet— Kötücül yazılımların tespiti Windows, Mac, Linux gibi geniş kitlelerin kullandığı işletim sistemleri de dahil olmak üzere, platformdan bağımsız bir biçimde karşımıza çıkmaktadır. Android işletim sistemi, akıllı telefonlarda pazar payı liderliği ve açık kaynaklı yapısıyla kötücül yazılımların birincil hedefi haline gelmiştir. Bunun sonucu olarak da bilişim suçlularının günümüzde öncelikli hedefi olan bu platform, aynı zamanda yeni güvenlik yöntemleri ve teknikleri tasarlayıp yeterliliklerini ölçmek isteyen araştırmacılar için de en öncelikli ortamlardan birini oluşturmaktadır. Olabildiğince fazla sayıda ve güncel çalışmaların incelenmesinin amaçlandığı bu literatür taramasında, Android işletim sistemini hedef alan kötücül yazılımların, yapay öğrenme teknikleri ve yaklaşımları kullanılarak tespit edilmesi konuları kapsanmıştır. Bu alandaki bilimsel araştırmalar mevcut literatürden derlenerek ilgili çalışmalarda kullanılan veya kullanılması önerilen tasarım, yöntem ve uygulamalar özet bir biçimde anlatılmıştır.

Anahtar Kelimeler— Android, kötücül yazılım tespiti, yapay öğrenme, veri madenciliği, öznitelik seçimi

Malware Detection on Android Platform: A Literature Review

Abstract— The problem of malware detection, irrespective of the platform of choice, affects nearly all operating systems, including the ones with large user bases such as Windows, Mac, and Linux. A significantly larger market share in the smartphone market compared to even its greatest rival and its open source architecture has made Android operating system the prime target for malware-related threats and cyber-attacks. Therefore, Android became the primary platform for designing and measuring the effectiveness of new approaches and methodologies for malware detection. This literature review focuses on the topic of detection of malware on Android platform by utilizing machine learning techniques and approaches. An extensive collection of the scientific studies on the given topic was collected and the design, the methodology and the real-world applications proposed or implemented by them are described in a short and concise manner.

Keywords— Android, malware detection, machine learning, data mining, feature selection

1. GİRİŞ (INTRODUCTION)

Android mobil işletim sisteminin, rakiplerine kıyasla sahip olduğu oldukça yüksek toplam pazar payının yanında toplamda sayısal olarak çok daha fazla uygulamaya sahip olması dolayısıyla kötücül / zararlı yazılımlar (malware) tarafından en sık hedef alınan platform olduğu bilinmektedir. Son kullanıcının güvenlik konularındaki bilinçsizliği ve yetersizliğine bağlı olarak, kötücül yazılımların Google Play Store vb. uygulama mağazalarında yayımlanmadan önce tespit edilmesi hayati bir öneme sahiptir. Bu kötücül yazılımların, bilgi güvenliğinin temel bileşenleri olan gizlilik, bütünlük ve kullanılabilirliğe saldıran ve bunlara zarar veren unsurlar

olduğu ve bunların sonucundan bireylerin ya da kurumların bilgi gizliliği başta olmak üzere, ciddi düzeyde çeşitli maddi ve / veya manevi zararlar verdiği bilinmektedir. Bu nedenlerle, Android destekli akıllı telefonlardaki uygulamalarda kötücül yazılımların hızlı, etkin, sürekli, verimli ve güvenilir bir şekilde belirlenmesi hem akademik, hem de ticari ve endüstriyel dünyada önemli bir konu olarak gündemde yer almaya devam etmektedir. Son yıllarda, Android vb. sistemlerde kötücül yazılım tespitinde yapay öğrenme (machine learning) algoritmaları ve yaklaşımları çok yaygın olarak kullanılmaya başlanmıştır. Bu araştırmanın amacı, Android işletim sistemini hedef alan kötücül yazılımların, yapay öğrenme yaklaşımları ile, bunlarla ilintili çeşitli istatistiksel ve veri madenciliği yöntemlerinin kullanılarak tespit edilmesi konularına ilişkin güncel çalışmaları kapsayan bir literatür incelemesi ortaya koymaktır. Farklı yöntem ve tekniklerle yapılan çalışmalar ve elde edilen performans ölçüm değerleri bir sonraki bölümde detaylı olarak ele alınmıştır. "Sonuçlar" bölümünde de literatürdeki bu çalışmaların farklı açılardan artıları ve eksileri, özellikle günlük yaşamda kullanılabilirliği ve ticari açıdan fizibiliteleri de irdelenerek değerlendirilmiş ve yorumlanmıştır.

2. İLGİLİ ÇALIŞMALAR (RELATED WORK)

Android işletim sisteminin güvenliğinin arttırılması yönündeki ilk bilimsel çalışma Linux işletim sistemi çekirdeğinde meydana gelen olayların izlenmesini temel alan bir model olup Schmidt ve arkadaşları [1] tarafından gerçekleştirilmiştir. Linux çekirdeğinden çekilen sistem çağrısı ve en son değiştirilen dosya isimleri gibi özellikler, temel düzeyde standart bir akıllı telefon kullanımı modelinin oluşturulması için kullanılmıştır. Android'in bu çalışma esnasında henüz bir işletim sistemi olarak piyasaya sürülmemiş olması, testlerin gerçek Android telefonlarda gerçekleştirilememesi sonucuna yol açmıştır. Aynı araştırmacılar daha sonraki bir çalışmalarında Android ortamında fonksiyon çağrılarını ayıklamak için APK (Android Package Kit) dosyalarının statik analizi için yeni bir yöntem önermiştir [2]. Önerdikleri yöntemde, fonksiyon çağrılarını sınıflandırmak için çağrı listeleri kötü amaçlı yazılım örnekleriyle karşılaştırılmıştır. Bu çalışmanın hemen sonrasında ise Enck ve arkadaşları tarafından Android izin (permission) modeli ve Android' in iç bileşenleri üzerine ilk kayda değer bilimsel çalışma ortaya konmuştur [3].

Fuchs ve arkadaşlarının [4] yaptıkları bir çalışmada Android uygulamalarının statik kod analizi ile Android tabanlı telefonlardaki kişisel verileri korumaya yönelik ScanDroid aracı geliştirilmiştir. SCanDroid, bildirim dosyasından (manifest file) güvenlik tabanlı özniteliklerin (features) çıkarılmasına dayanır ve çalışan uygulamalar arasında gerçekleştirilen tüm veri akışlarının tanımlanmış güvenlik kurallarıyla uyumlu olup olmadığını kontrol ederek olası kişisel veri ihlallerinin tespiti durumunda alarm vermeyi amaçlar.

Bläsing ve arkadaşları [5] Android uygulaması kum havuzunu (sandbox) kullanan bir dinamik analiz yöntemi önermiştir. Kötücül örüntüleri (pattern) tespit etmek için, Android APK dosyalarının statik analizi yapıldıktan sonra, uygulamaların dinamik analizi için dosyalar kum havuzunda çalıştırılmıştır. Kum havuzundaki söz konusu dosyanın çalıştırılması esnasında, açılan, erişilen, değiştirilen dosyalar gibi eylemler izlenmektedir. "ADB Monkey" adı verilen kullanıcı etkileşimini simüle eden bir uygulama kullanılmış olması, simüle edilen kullanıcı etkileşiminin gerçek dünyadaki kullanıcı etkileşimini aynı biçimde göstermeyebileceği gerçeği bu çalışmanın başlıca eksikliği olarak görülebilir.

Wu ve arkadaşları, yetki ve izinler, bileşenlerin dağıtımı (deployment of components), haberleşme (intent) mesajları ve UPA (Uygulama Programlama Ara yüzü) çağrıları gibi öznitelik türlerini kullanan DroidMat'ı geliştirmiştir [6]. Kötücül yazılım örneklerini "Contagio Mobile" sitesinden [7] toplamış ve zararsız (benign) uygulama örneklerini de Google Play mağazasından edinmişlerdir. Veri kümelerinde 238 adet kötücül yazılım (malware) ve 1500 adet zararsız uygulama örneği vardır. Sınıflandırmak için çeşitli tiplerde kümeleme (clustering) algoritmalarından yararlanmışlar ve gerçekleştirdikleri testlerde %97.87 oranında doğruluk (accuracy) oranı elde etmişlerdir.

DroidAPIMiner'ı geliştiren araştırmacılar, APK dosyalarının taşınabilir kod (bytecode) gösterimi içindeki UPA düzeyinde öznitelikleri elde etmeye dayanan bir yöntem ortaya koymuştur [8]. Yöntemlerinde, öznitelik olarak izin / yetki istemleri, kritik UPA çağrıları, paket seviyesi bilgileri ve uygulama parametreleri kullanılmıştır. DroidAPIMiner, Androguard kütüphaneleri üzerine kurulmuştur ve RapidMiner adlı veri madenciliği yazılımı kullanılarak yapay öğrenme modelleri oluşturulmuştur. Veri kümeleri üç farklı kaynaktan toplanmış olup kötücül yazılım örnekleri için McAfee uygulama arşivlerinden ve MalGenome projesinden, zararsız uygulama örnekleri için de Google Play mağazasından yararlanılmıştır [8]. 3987 adedi kötü amaçlı yazılım olup geri kalanı da Google Play mağazasından elde edilmiş olan toplamda yaklaşık 20,000 adet uygulama analiz edilmiştir. Çalışmada ID5 Karar Ağacı [9], C4.5 Karar Ağacı [10], k-NN [11] ve destek vektör makinesi (support vector machine) [12] yapay öğrenme algoritmalarından yararlanılmıştır. En yakın k komşu (k-NN) algoritması kullanılarak testlerde %99 doğruluk ve %2.2 yanlış pozitif oranı elde edilmiştir.

Android için bir bilgi akışı izleme aracı olan TaintDroid dinamik iz (taint) izleme kabiliyeti sağlamaktadır [13]. Yüksek düzeyde görünürlük ve kontrol sağlamak için bir uygulamanın özel verileri nasıl kullandığı aynı anda birden fazla kaynaktan izlenmektedir. Android'in sanallaştırılmış ortamı kullanılarak uygulamaların gerçek zamanlı analizi gerçekleştirilmiştir. Birçok özel veri kaynağı, dört farklı izleme düzeyine göre etiketlenmiştir. Bunlar değişken düzeyi, metot düzeyi, dosya düzeyi ve mesaj düzeyidir. Saldırı belirleme sistemindeki (intrusion detection system) çalışma ilkesine benzer şekilde, koruma altındaki sistemde etiketlenmiş olan verilerin, tanımsız / güvensiz üçüncü şahıs bir uygulama kullanılarak iletimi durumunda uyarı / alarm verilir.

Android sistemlerindeki zararlı kodlara ilişkin olarak, veri iletimi ve haberleşme sırasındaki riskleri analiz etme ve ölçmeye yönelik bir çalışma yapılmıştır [14]. Bu çalışmada, olasılık temelli yapay öğrenme yöntemleri kullanılarak ve Android izinlerine bakılarak risk düzeyi ve risk puanlarının hesaplanması önerilmiştir. Testlerde elde edilen sonuçlarda olasılık tabanlı genel modellerin, bilinen diğer yöntemlerden önemli ölçüde daha başarılı olduğu gözlemlenmiştir. Bunun yanı sıra, Naive Bayes algoritmasına dayanan yapay öğrenme modellerinin de kayda değer bir risk ölçme ve değerlendirme alternatifi olabileceği vurgulanmıştır [14].

Droidbox, TaintDroid'in mevcut işlevlerine Android UPA izleme ve dosya sistemindeki raporlama işlevselliği ekleyerek geliştirilmiştir [15]. Bu uygulama, izlenen aktivitenin bir zaman çizelgesi görünümünü kullanıcıya sunarak izlenen davranışsal özelliklere göre kötücül yazılımın ayırt edilmesine olanak sağlamaktadır.

ComDroid projesinde, Android uygulamalarının kaynak koduna geri derlenmiş kodlarının statik analizini yaparak zayıf izinlerle gönderilen Android haberleşme nesnelerini bulmaya dayanan bir yöntem ortaya konulmuştur [16].

Burguera ve arkadaşları tarafından önerilen Crowdroid ise, dinamik analiz yapan yapay öğrenme temelli bir sistemdir [17]. Crowdroid, Android yüklü telefonlarda bir uygulamanın çalışması sırasında yaptığı kullanıcı etkileşimi gerektiren sistem çağrı sayısına bakarak Truva Atı (Trojan Horse) benzeri kötücül yazılımları tespit etmektedir. İzlenen özellikler bulut sisteminde analiz edilerek yapılandırılmakta ve K-ortalama (K-means) algoritması ile sınıflandırılmaktadır. Araştırmacıların kendilerinin geliştirdiği kötücül amaçlı yazılım örneklerinde %100'lük bir belirleme / tanıma oranına ulaşıldığı ifade edilmektedir [17].

Yerima ve arkadaşları [18] statik kod analizinden elde edilmiş öznitelikleri ve parametreleri kullanarak Naive Bayes algoritması ile yapay öğrenme yapan bir yaklaşım ortaya koymuştur. Android'de yaygın olan kötücül uygulama örnekleriyle test edildiğinde %90.60 düzeyinde bir belirleme oranına ulaştıkları belirtilmiştir.

Yüksek düzeyde doğruluk (accuracy) skoru sağlayan bir sınıflandırma yöntemi Elish ve arkadaşları tarafından yapılan bir çalışmada ortaya konmuştur [19]. Kullanıcı girdilerinin, uygulamanın çalıştırılması sırasında yapılan UPA çağrılarını nasıl tetiklediği üzerine bir veri akışı özelliğinin statik olarak çıkarılmasına dayanmaktadır. 1433 adet kötücül kod örneği ve 2684 adet zararsız kod örneği üzerinde değerlendirdikleri bu yöntem ile %2.1 yanlış negatif ve %2 yanlış pozitif oranlarını elde etmişlerdir [19].

Android yüklü telefonlardaki kötü amaçlı yazılım tespiti üzerine 2014 yılında geniş kapsamlı bir araştırma gerçekleştirilmiştir [20]. "Drebin" adını verdikleri bir uygulama geliştiren araştırmacılar, akıllı telefon sistem kaynaklarının hızlı bir şekilde harcanmasını önlemek amacıyla, kötü amaçlı yazılım tespiti için kapsamlı bir statik analiz yapılmasını önermiştir. Bu nedenle de, APK dosyalarından çıkarılacak öznitelik türünün olabildiğince geniş tutulmasına karar verilmiştir. Dosya ve uygulama izinleri, UPA çağrıları ve ağ adresleri gibi öznitelikleri içeren toplamda 8 farklı öznitelik türü belirlenmiştir. Bütün bu öznitelikler daha sonra kötücül haberleşme nesnelerini otomatik olarak belirleme amacıyla bir birleşik vektör uzayında (joint vector space) birleştirilmiştir. 5560 adet kötücül yazılım ve 123,453 adet zararsız yazılım örneği kullanarak çeşitli deneyler yapılmıştır [20]. Yapay öğrenme ile sınıflandırma aşamasında DVM (destek vektör makinesi) algoritması kullanılarak %94'lük kötücül yazılım tespit başarısına ve %1'lik yanlış pozitif (false positive) oranına ulaşılmıştır.

Sanz ve arkadaşları karakter dizisi (string) analizi ile kötücül yazılım tespiti için bir yöntem geliştirerek Android APK dosyalarından karakter dizisi özelliğini elde etmiştir [21]. Gerçekleştirdikleri testlerde 333 kötü amaçlı yazılım örneği ve 333 zararsız uygulama örneği barındıran bir veri seti kullanılmış ve her bir farklı Android uygulamasını ayırt etmek için metin madenciliği tekniklerinden faydalanılmıştır. Naive Bayes, Bayes Ağları [22], DVM, k-NN, J48 ve Random Forest algoritmalarını WEKA yapay öğrenme platformunda çalıştırıp k katlamalı çapraz doğrulama (k-fold cross-validation) yöntemiyle test sonuçlarını elde etmişlerdir. En yüksek doğruluk oranı %94.70 düzeyinde gözlemlenmiştir [21].

Bir başka çalışmada ise, araştırmacılar veri akışı UPA'larını öznitelik olarak kullanan bir yaklaşım ile Android'deki kötücül yazılımların tespitini önermiştir [23]. UPA düzeyinde öznitelikler elde etmek ve k-NN

sınıflandırma modelini geliştirmek için kapsamlı bir analiz yapılmıştır. Bütün veri madenciliği süreçlerinde sadece "veri akışı UPA" öznitelik türü kullanılmıştır. Veri akışıyla bağlantılı UPA listesini optimize ederek hassas veri iletimi analizinin etkinliğinin önemli ölçüde arttığı gözlemlenmiştir. Kullandıkları veri kümeleri 1160 adet zararsız ve 1050 adet kötü amaçlı yazılım örneği içermekte olup gerçekleştirdikleri deneylerde %97.66'ya varan doğruluk oranları elde etmişlerdir.

Gerçek zamanlı bir kötü amaçlı yazılım tespit sistemi olan TStructDroid zaman serisi öznitelik kayıtlama, bölütlere ayırma ve frekans bilgisi çıkarımı gibi yöntemleri kullanmaktadır [24]. Uygulama çalıştırma izlerindeki gizli örüntüleri çıkarmak için yeni bir bilgi işleme yapısı önerilmiştir. Bu yöntemde, J48 karar ağacı tabanlı bir sınıflandırıcı kullanılmıştır. Standart çapraz doğrulama testlerinde, %90 ile %93.6 arasında doğruluk ve yanlış pozitif oranı olarak da %5.4 ile %7.3 arasında sonuçlar elde etmişlerdir. Sadece 110 zararsız ve 110 kötücül yazılımı kullandıkları gerçek zamanlı canlı testlerde elde ettikleri %98 doğruluk ve %1'den daha az yanlış pozitif oranıyla da kayda değer sonuçlara ulaştıklarını belirtmişlerdir [24].

DroidScope, Android platformunda kötücül eylemleri ve gizli bilgilerin sızdırılmasını tespit etmek için dinamik profil oluşturan ve bilgi izleme yapan çok düzeyli bir anlamsal analiz aracıdır [25]. Komut izleme ve UPA çağrı izleme gerçekleştirir ve potansiyel hassas bilgi sızıntılarını keşfetmek için leke analizi yöntemleri kullanır.

Dini ve arkadaşları [26] Android kötü amaçlı yazılım tehditlerine karşı, hem işletim sistemi çekirdeği hem de kullanıcı uygulama katmanında faaliyet gösteren, çok düzeyli bir aykırılık belirleme (anomaly detection) tekniği olarak MADAM'ı geliştirmiştir. Bu sistemin çok düzeyli olması, daha önce görülmemiş kötücül yazılımların tespit edilmesini sağlayabilecek zengin bir öznitelik alanının saptanmasını olanaklı kılmaktadır. Sistem sırasıyla eğitim, öğrenme ve çalışma aşamalarını kapsamakta ve sınıflandırma için k-NN algoritması kullanılmaktadır. Çalışma sırasında eğitim ve öğrenme kümelerine yeni öznitelik vektörleri ekleyerek, yapay öğrenme tabanlı modelin kötü amaçlı yazılım tespit oranlarını iyileştirmeyi amaçlamaktadır. 10 kötü amaçlı yazılım ve 50 zararsız örneği içeren veri setinin kullanıldığı deneylerde ortalama %93 doğruluğa ve %5 yanlış pozitif oranlarına ulaşılmıştır [26].

Smartdroid, Zheng ve arkadaşları tarafından geliştirilmiş olup kullanıcının ara yüzle olan etkileşimini izleyen karma yapıda otomatik bir kötücül yazılım algılama uygulamasıdır [27]. Aktivite kontrol grafikleri ve fonksiyon çağrısı grafikleri statik analiz sırasında bir yol seçici tarafından oluşturulur. Kullanıcı ile etkileşim sonrası oluşturulan yeni aktiviteleri sınırlandırmak için fonksiyon çağrısı grafikleri dolaylı olarak veya olay tabanlı UPA çağrılarına bağlı olarak güncellenir. Dinamik analiz için, Android çerçeve kodu değiştirilir ve kullanıcı ara yüzü ile etkileşime girdikten sonra oluşturulan yeni aktiviteleri sınırlandırmak için kısıtlayıcı bir bileşen eklenir.

Yerima ve arkadaşları [28] Android'de kötücül yazılım tespiti için kolektif öğrenme (ensemble learning) yöntemleri kullanan bir yaklaşım geliştirmiştir. Statik analizi, kolektif yapay öğrenme yöntemi kullanmanın verimlilik ve performans avantajlarıyla birleştirerek Android kötücül yazılım tespitinde doğruluk oranının yükseltilmesi amaçlanmıştır. Testlerde McAfee'nin uygulama deposundan toplanan 2925 kötü amaçlı yazılım ve 3938 zararsız uygulama örneği kullanılmıştır. Önerdikleri yöntemde, kolektif öğrenme tekniklerinden daha çok katkı alabilmek amacıyla geniş bir öznitelik alanı kullanılmış ve %97.3 ile %99 arasında değişen doğruluk oranlarına ulaşılmıştır [28].

Mlifdect'in geliştiricileri [29] ilgili makalelerinde geleneksel yapay öğrenmeye dayalı kötü amaçlı yazılım tespit yöntemlerinin, tekli sınıflandırma algoritmalarının tercih edilmesi dolayısıyla ancak sınırlı bir tespit doğruluğuna ulaşabildiğini iddia etmiştir. Kötücül yazılım tespitinde başarıyı arttırmak için paralel yapay öğrenme ve bilgi birleştirme tekniklerinden yararlanan yeni bir yaklaşım ortaya konulmuştur. Bu yöntemde izinler, UPA çağrıları ve bileşenlerin yerleşimi de dahil olmak üzere sekiz tür öznitelik statik analiz yöntemiyle APK dosyalarından çıkarılmaktadır [29]. Sınıflandırma sürecini hızlandırmak amacıyla paralel çalışan bir yapay öğrenme modeli geliştirmişlerdir. 3982 adedi kötücül uygulamalar, geri kalanı ise zararsız örneklerden oluşan bir veri kümesi kullanılmıştır [29]. Kötücül uygulama örnekleri için Drebin [20] ve MalGenome [30] projelerinden yararlanılmış, zararsız örneklerse Google Play Store'dan indirilmiştir. Deneyler sonucunda zararsız ve zararlı yazılım uygulamaları belirlemede %99.7 doğruluk oranı elde edilmiştir [29].

Bir başka çalışmada, Android'deki bazı kötücül yazılımların dinamik analizlerinin benzetici (emulator) kısıtlarından kaynaklı sorunlarını gidermek amacıyla otomatik öznitelik çıkarımında gerçek telefonlar kullanılmıştır [31]. Dinamik öznitelikler Android telefonlar üzerinde çalıştırılan uygulama örneklerinden otomatik olarak çıkarılmıştır. Bunun yanı sıra, gerçekleştirdikleri bazı testlerde benzetici tabanlı ve cihaz tabanlı kötü amaçlı yazılım tespit yöntemlerinin karşılaştırmalı analizleri yapılmıştır. Dinamik ve davranışsal analiz temelli çalışmaların çok büyük bir kısmında benzetici kullanıldığı için, Alzaylaee ve

arkadaşlarının bu çalışması ayrı bir önem taşımaktadır. MalGenome projesinden elde edilen 222 adet kötü amaçlı yazılım örneği ve Intel Security' e bağlı McAfee Labs'den elde edilen 1222 adet zararsız örneği kullanılmıştır [31]. Test ortamı olarak Santoku Linux Sanal Cihazı [32] tabanlı bir makinenin yarattığı "Android emulator" ortamı yanı sıra gerçek bir akıllı telefon kullanılmıştır. Testlerde WEKA veri madenciliği platformu kullanılmış ve elde edilen 178 öznitelikten 100'ü WEKA' daki bilgi kazanımı (information gain) öznitelik seçme algoritmasının sıralaması temel alınarak seçilmiştir. WEKA'da bulunan DVM, Naive Bayes, basit doğrusal regresyon, çok katmanlı yapay sinir ağı, kısmi karar ağaçları (Partial Decision Trees), Random Forest ve J48 karar ağacı algoritmalarını kullanmışlardır. Cihaz üzerinde gerçekleştirilen test sonuçları ile benzetici üzerinde gerçekleştirdikleri testlerin sonuçlarını karşılaştırmışlar ve elde ettikleri bu sonuçları önceki benzer çalışmaların [33, 34] sonuçlarıyla da karşılaştırmışlardır. Cihazlardaki testlerde %93.1 gerçek pozitif oran ile birlikte 0.926 F1 skoru ölçümüne ulaşılmış, öte yandan Random Forest sınıflandırıcısının kullanıldığı testlerde ise %14.85 yanlış pozitif oranı gibi yetersiz sonuçlar da alınmıştır. Benzetici ortamında gerçekleştirilen deneylere kıyasla cihaz üzerinde yapılan testlerde genel olarak daha iyi tespit oranları elde edildiği gözlemlenmiştir [31].

Yerima ve arkadaşları, Android kötü amaçlı yazılımların erken uyarı ve tespiti amaçlı bir paralel yapay öğrenme kullanımına dayanan bir sınıflandırma yaklaşımı önermiştir [35]. Yapay öğrenme modelinin eğitim aşamasında APK dosyalarından çıkarılan üç tip öznitelik kullanılmıştır. Bunlar, UPA ile ilgili özellikler, uygulama izinleri, standart işletim sistemi ve Android mimarisi komutlarıdır. Testlerde kullanılan yapay öğrenme algoritmaları C4.5 karar ağacı, basit doğrusal regresyon, Naive Bayes ve ayrıca PART ve RIDOR (Ripple-Down Rule) adlı kural tabanlı sınıflandırıcılardır. 125 adet UPA ile birlikte 54 adet UPA çağrısı ile ilgili öznitelikler çıkarılmıştır. McAfee'nin uygulama arşivinden 2925 adet zararlı ve 3938 adet zararsız olmak üzere toplam 6863 Android uygulaması edinilmiştir. 39 farklı yapay öğrenme algoritması içerisinde PART algoritması ile en başarılı sonuç olan %95.8 tespit oranı elde edilmiştir [35].

Yapay öğrenme kullanarak kötü amaçlı yazılım tespiti bağlamında izinler özelliğini kullanan bir başka yaklaşım Peiravian ve Zhu [36] tarafından önerilmiştir. Testlerde destek vektör makineleri ve karar ağaçları gibi çeşitli sınıflandırıcılar kullanılmıştır. Her biri aynı sayıda örnek, ancak farklı sayıda öznitelik içeren üç temel veri seti oluşturulmuştur. Bunlardan 1260 adedi kötü amaçlı yazılım, geri kalan 1250 adedi ise zararsız olan toplamda 2510 örnek toplanmıştır. Veri kümelerindeki özdeş öznitelik değerlerine sahip kötücül yazılımları elemişler ve MalGenome veri kümesinin 49 farklı kötü amaçlı yazılım ailesinden 1250 kötü amaçlı yazılımdan sadece 610 kötü amaçlı yazılım örneğini kullanmışlardır. Zararsız uygulamalar Google Play mağazasındaki 25 farklı uygulama kategorisinden indirilmiştir. Çalışmada DVM algoritmasını kullanarak %96.88 doğruluk yüzdesi ve %94.8 hassasiyet (recall) yüzdesiyle en yüksek tespit oranlarına ulaşmışlardır [36].

Alatwi ve arkadaşları, kategori tabanlı yapay öğrenme sınıflandırıcılarının, kategoriye dayalı olmayan sınıflandırıcılara kıyasla oldukça yüksek tespit oranlarına olanak sağladığını öne sürmüştür [37]. Önerdikleri yöntemde her farklı kategori için bir kötü amaçlı yazılım tespit sınıflandırıcısı belirlenmiştir. Uygulamaların öznitelikleri ile ait olduğu kategorinin gerektirdiği işlevselliği sağlamak için gereken öznitelikler arasında bağlantı kurulmuştur. APK dosyaları kendi JAVA kaynak kodlarına göre tersine mühendislikle çevirmişler ve izinler, yayın alıcıları (broadcast receivers) ve UPA çağrıları gibi öznitelikleri çıkartmışlardır. Toplamda üç veri seti hazırlanmış ve her veri setinin %70'i eğitim, %30'u ise test için kullanılmıştır. Veri kümeleri, testlerinde kullandıkları yapay öğrenme sınıflandırıcıların performans ortalamasını almak için gerçekleştirdikleri 50 tekrarın her turunda rasgele biçimde karıştırılmıştır. Gerçekleştirdikleri testlerde kategori tabanlı ve kategori tabanlı olmayan DVM sınıflandırıcıları kullanılmış ve elde ettikleri sonuçlar karşılaştırılmıştır. Kategori tabanlı sınıflandırıcı ortalama %98.72'lik doğruluk oranına ulaşılırken, kategori tabanlı olmayan sınıflandırıcının ise ortalama %94.58'lik doğruluk oranına ulaştığı gözlemlenmiştir [37].

Mariconti ve arkadaşları [38] tarafından yayınlanan çalışmada ise araştırmacılar, uygulamaların çalışma tarzlarını inceleyen MAMADROID sistemini tasarlamıştır. MAMADROID, UPA çağrıları dizisinden Markov zinciri olarak soyutlanmış bir davranışsal model oluşturarak, bu modeli gereksinim duydukları öznitelikleri çıkarmakta ve kötü amaçlı yazılım tespiti için kullanmaktadır. Önerdikleri yöntemin performansını, altı yıl içerisinde toplanan 8500 zararsız ve 35,500 zararlı yazılım uygulama örneği içeren veri seti üzerinde test etmişlerdir. MAMADROID'in diğer model ve yöntemlerden farkı, UPA çağrılarının kullanımı veya sıklığı yerine uygulama tarafından gerçekleştirilen soyutlanmış UPA çağrıları sırasına dayanan bir sistem olmasıdır. Bir uygulama tarafından gerçekleştirilen UPA çağrıları arasındaki geçişleri temsil eden istatistiksel bir model oluşturularak, bu geçişler Markov zinciri olarak modellenmiş ve bu model sayesinde öznitelikler seçilip yapay öğrenme algoritmaları ile veriler sınıflandırılmıştır [38]. Sınıflandırma aşamasında RandomForest, 1-NN, 3-NN ve

DVM olmak üzere dört farklı yapay öğrenme algoritmasından yararlanılmıştır. Testlerde en yüksek F1 skoru (F1 score, F-measure) %99 olarak elde edilmiş, ayrıca MAMADROID'in testleri boyunca geçen üç yıllık süre göz önüne alındığında bu modelin oldukça başarılı bir tespit oranını uzun vadede koruduğu da ortaya konulmuştur.

Onwuzurike ve arkadaşları [39] tarafından yayınlanan daha yeni bir çalışmada ise, MAMADROID ile benzer bir modelleme yaklaşımı kullanılarak statik, dinamik ve melez analiz yöntemlerini birlikte kullanan bir yapı önerilmiştir. MAMADROID'i dinamik analiz yapısına dönüştürebilmek için, CHIMP [40] sanal cihazı üzerinde değişiklikler yapılmıştır. CHIMP sanal cihazında uygulama yürütülürken oluşturulan izlerden (trace) UPA çağrıları elde edilmiştir. Geliştirilen sistem AUNTIEDROID olarak adlandırılmış, hem sahte rastgele (pseudorandom) girdiler (inputs), hem de gerçek kullanıcı girdileri kullanılmıştır [39]. AUNTIEDROID'in MAMADROID'den temel farkı ise, statik analiz yerine dinamik analiz yapılarak çıkarılan davranışsal özniteliklere dayanmasıdır. Testlerinde kullandıkları veri kümeleri için araştırmacılar, "zararsız ve yeni" olarak etiketlenmiş 2568 adet zararsız uygulama setini [38] ve VirusShare'den [41] elde edilmiş 2692 adet Android kötü amaçlı yazılım örneğini kullanmışlardır. Araştırmacılar, statik ve dinamik analiz tekniklerinin bir arada kullanılmasının en iyi sonucu verdiğini ve %92 oranında bir F1 skoru değerine ulaştıklarını, bunun yanı sıra statik analiz tekniklerinin en az dinamik analiz teknikleri kadar etkili olduğunu vurgulamıştır [39].

Normalize edilmiş Bernoulli Naïve Bayes adı verilen değişik bir Naive Bayes algoritması, kötücül Android uygulamalarını tespit etmek için geliştirilmiştir [42]. Kullanılan 13 öznitelik tipinin büyük bir kısmı, "AndroidManifest.xml", "classes.dex" ve "resources.arsc" isimli üç Android dosyasından elde edilmiştir. Deneylerde F-Secure'dan zararlı veya zararsız olarak etiketlenmiş 120,000'in üzerinde uygulama örneği kullanılmıştır. Veri kümelerinden seçilen 10,000 eğitim ve 10,000 test örneği kullanılarak %82.10 gerçek pozitif oranı (true positive rate) ve %0.1 yanlış pozitif oranlarına ulaşılmıştır [42].

Bir başka çalışmada araştırmacılar uygulama kaynak kodunu imza olarak değerlendirerek uygulamalardaki zayıflıkları açığa çıkarmak amacıyla "n-gram" yöntemini kullanmıştır [43]. APK dosyalarından UPA çağrısı, çağrı akışı (call flow) ve cihaz belleği (device memory) öznitelik türleri elde edilmiştir. Sistem tarafından gerçekleştirilen temel işlev ise, test edilmek istenen uygulamanın Google Play Store gibi bir uygulama mağazasında yayınlanmasından önce, statik analizle elde edilen n-gram imzasının daha önce kaydedilmiş olan imzalarla karşılaştırılması ile potansiyel güvenlik zayıflıklarının veya kötücül yazılımların belirlenmesidir.

MARVIN adı verilen statik ve dinamik analizi birleştiren karma bir yaklaşım daha önerilmiştir [44]. MARVIN, her bir Android uygulaması için bir kötü niyetlilik / zararlılık puanı üreterek, risk değerlendirmesi temelli bir yapay öğrenme kullanan bir kötü amaçlı yazılım tespiti uygulamasıdır. Mobil aygıtın dışındaki bir platformda APK dosyalarının hem statik hem de dinamik analizini gerçekleştirmişler ve her bir uygulamanın davranışsal özelliklerini belirlemek amacıyla kapsamlı bir öznitelik seti kullanmışlardır. Öznitelik seçimi için F1 skoru ölçümünden yararlanılmıştır. Veri kümeleri 15,000 adedi kötü amaçlı yazılım olmak üzere toplamda 135,000 Android uygulamasından oluşmuştur. Deneylerde %98.24'lük tespit oranı ve %0.04'ten daha düşük bir yanlış pozitif oranı elde edilmiştir [44].

Cihaz üzerinde gerçek zamanlı yapay öğrenme teknikleri kullanan bir diğer çalışmada [45] ise, araştırmacılar Android kötü amaçlı yazılımlarından davranışsal öznitelikleri çıkarmak için Cuckoo Sandbox [46] dosya analiz platformundan elde ettikleri raporlardan yararlanmıştır. Kötücül yazılım örneklerini VirusShare [41] adlı kötü amaçlı yazılım paylaşım platformundan elde etmişler ve geliştirdikleri cihaz üzerinde yapay öğrenme algoritmasını 18 farklı kötü amaçlı yazılım ailesine ait toplamda 2000 örnek üzerinde test etmişlerdir. 10 katlamalı çapraz doğrulama (10-fold-cross-validation) yöntemi kullanılarak elde edilen doğruluk oranı 0.89 olmuştur [45].

Bir başka çalışmada, kötücül Android uygulamalarının statik analizi için iki farklı yapay öğrenme yaklaşımına dayanan bir yöntem ortaya konmuştur [47]. Bu iki yaklaşımdan ilki, uygulama izinlerini öznitelik olarak kullanmaya dayanırken, diğeri ise sözcük torbası (bag of words) modelini kullanarak kaynak kodun analizine dayanır. Yapay öğrenme modellerinin eğitimi ve test edilmesi için 200 adet kötü amaçlı yazılım ve 200 adet zararsız örnek içeren M0Droid [48] veri setini kullanmışlardır. Weka 3.6.6 kütüphanesi kullanılmış ve OWASP Seraphimdroid Projesi'nde kullanılan tarayıcıya (browser) entegre edilmiştir [49]. Kaynak kodu analizi temelli yaklaşımda önce APK dosyaları tersine mühendislikle kaynak kodlarına geri çevrilmekte ve elde edilen bu kaynak kodları, sözcük torbası tekniğine dayanan metin madenciliği yöntemi ile birlikte kullanılmaktadır. APK dosyalarından elde edilen kaynak kodları, kelimelerden oluşan torbalardan (bags) oluşan tekli sözcüklere (unigram) dönüştürülmektedir. Gerçekleştirdikleri testlerde J48, Naive Bayes, DVM,

Random Forest, JRIP, Lojistik Regresyon ve AdaBoostM1 algoritmaları kullanılmıştır. Kaynak kodu temelli yaklaşımlarında %95.1 F1 skoru, izin tabanlı yaklaşımlarında ise %89'luk bir F1 skoru elde edilmiştir [47].

Dendroid adıyla geliştirilen bir sistemle, akıllı telefonlardaki kötücül uygulamaların kaynak kodlarındaki belirli kod yapılarına göre otomatik şekilde analiz etmeye dayalı yeni bir yöntem ortaya konmuştur [50]. Kod yapıları, uygulama örneklerinde mevcut olan her fonksiyonun kontrol akış grafiğini (control flow graph) temsil etmektedir. Araştırmacılar, testlerde kullandıkları veri kümelerini MalGenome projesindeki örneklerden alt kümeler seçerek oluşturmuşlar ve 33 farklı kötücül yazılım türünde gruplanmış olan 1231 adet kötücül uygulama örneğine indirgemişlerdir [50]. Klasik metin madenciliği yöntem ve uygulamalarında izlenen modelleme sürecini yeniden düzenleyerek test örneklerini, ait oldukları kötücül yazılım gruplarına atamak için kullanılan eğitim veri kümelerindeki örneklerle karşılaştırmış ve aralarındaki benzerlik oranını ölçmüşlerdir [50]. Bunun yanı sıra, her bir kötü amaçlı yazılım ailesinden elde edilen öznitelik vektörlerini kullanan bir hiyerarşik kümeleme uygulaması üzerinde de çalışılmıştır. Ortaya koydukları yöntemin uygulaması sonucu ortaya çıkan öbek ağaçlarının (dendrogram), biyolojik türler için Filo genetik ağaçlara benzediği gözlemi, kötü amaçlı yazılım aileleri arasındaki evrimsel ilişkileri belirleme yönünde yol almalarını sağlamıştır. Testlerden elde dilen sonuçlarda oldukça yüksek düzeyde doğruluk oranlarına ulaşıldığı belirtilmiştir [50].

Coronado-De-Alba ve arkadaşları tarafından statik analiz yöntemleri kullanan bir üst-kolektif (meta-ensemble) tabanlı bir yapay öğrenme yöntemi önerilmiştir [51]. Testlerde Drebin'den elde edilmiş 1531 adet kötü amaçlı yazılım ve Google Play mağazasından indirilmiş 1531 adet zararsız uygulama örneği kullanılmıştır. Veri kümesindeki toplam 3062 adet örnekten izinler, haberleşme nesneleri, donanım ve yazılımdan seçilen toplam 660 öznitelik elde edilmiştir. Dengeli ve dengesiz veri kümelerinde Ki-Kare (Chi-Square) ve Relief adlı öznitelik seçimi algoritmalarını kullanmışlardır. WEKA'da her bir yapay öğrenme algoritması için varsayılan parametre ayarları kullanılarak birçok farklı sınıflandırıcıyı çalıştırmışlardır. Test sonuçlarını elde etmek için 10 katlamalı çapraz doğrulama yöntemi kullanılmıştır. Bir kolektif öğrenme sınıflandırıcısı olan RandomForest algoritmasını kullanarak %97.56 düzeyinde tespit başarısına ulaştıklarını paylaşmışladır [51].

Android sistemleri için bugüne kadar yapılmış olan en geniş kapsamlı zararlı kod araştırması, 2010 yılından 2017 yılına kadar toplanan 1200 adet kötü amaçlı farklı yazılım gruplarına ait 1.28 milyondan fazla kötücül kod örneğinin analiz edildiği bir çalışma olarak bilinmektedir [52]. Çalışmanın bu denli geniş çaplı olması nedeniyle, veri setinin otomatik analizini yapabilmek için özel yazılımlar geliştirilmek zorunda kalınmıştır. Yeniden paketleme (re-packing) tabanlı kötücül yazılım türlerine odaklanılarak, bu gibi yazılımların çalışma tarzları ve biçimlerinin 2010'dan 2017'ye kadar nasıl ve ne yönde geliştiği anlaşılmaya çalışılmıştır. Paketleme temelli kötücül yazılımlar, zararsız uygulamalara "rider" olarak adlandırılan kötücül uygulama eşlikçisi (payload) eklemektedir. Araştırmacılar, uygulamanın kötücül kısmını oluşturan eşlikçiyi zararsız kısımdan ayırt etme amacıyla diferansiyel analiz yöntemini uygulamışlardır [52]. Testlerinde kullandıkları zararlı ve zararsız uygulama örneklerini AndroZoo'dan [53] elde etmişlerdir. Rider'ları çıkarmanın sistematik bir yolunu oluşturmak için, aynı kötü amaçlı yazılım ailesinin üyeleri arasında ortak olan öznitelikler elde edilmiştir. Aynı aileden örnekler arasında hangi özniteliklerin yaygın olduğunu belirlemek için Dendroid'den [50] yararlanılmıştır. Dendroid'i, DEX (Dalvik çalıştırılabilir dosyası) veya APK uygulaması dahilindeki mevcut tüm kaynaklardan öznitelikleri yinelemeli (recursive) biçimde çıkarmak için kullanmışlardır. Veri kümelerindeki 1.2 milyon örnekten toplamda 155.7 milyon adet metot incelenmiş ve bunlardan 1.3 milyon adedin Rider metodu / fonksiyonu olduğu tespit edilmiştir. Bu kapsamlı araştırma, kötücül Android yazılımlarının 2010'dan 2017'ye kadar hızla gelişip evrimleştiğini ortaya koymuş, ayrıca bu değişikliklere aynı hızla ve başarıyla ayak uydurabilecek güvenlik sistemlerinin geliştirilmesinin önemini kanıtlamıştır [52].

Yakın zamanlarda yapılan bir başka çalışmada değişik bir yöntem kullanılarak MalDozer modeli geliştirilmiştir [54]. MalDozer'da, kötücül Android uygulamalarının sınıflandırılması ve tespiti için derin öğrenme [55] tekniklerinden ve modellerinden yararlanılmıştır. Çalışmada UPA çağrıları için yapay sinir ağlarını [56] kullanan MalDozer, bir uygulamanın UPA çağrılarının sınıflamasını otomatik olarak yapmakta, kötücül uygulamaların tespiti için test verisinde bulunan zararlı ve zararsız örüntüleri derin öğrenme yaklaşımı ile öğrenmektedir. MalDozer, Google Play mağazasından indirilen 38,000 zararsız uygulamanın yanı sıra 20,000 zararsız örnek ve 33,000 kötü amaçlı yazılım örneğini içeren bir birleştirilmiş veri seti üzerinde test edilmiştir. Bu veri setinde yapılan testlerde, MalDozer'in, %96 ile %99 arasında F1 skoru elde ettiği gözlemlenmiştir. [54].

Yapay öğrenme destekli bir başka güncel çalışmada da, 200 adet zararlı ve 200 adet zararsız Android uygulamasının kaynak kodları analiz edilmiş ve %95.1

düzeyinde F1 skoru elde edilmiştir [57]. 2017 yılında yapılan bir başka çalışmada ise, ilgili literatürdeki tüm çalışmalardan daha farklı bir yöntem izlenmiş ve zararlı uygulama paketlerinin belirlenmesinde Android uygulamalarındaki bindirmeli (piggybacked) erişimlerden yararlanılmıştır [58].

En güncel çalışmalardan birisi de EveDroid modelidir [59]. EveDroid, Nesnelerin İnterneti (Internet of Things) kapsamında kullanılan Android işletim sistemli aygıtlar için kötü amaçlı yazılım algılama aracı ve modelidir. Uygulama programlama ara yüzü çağrılarını (API calls) parametrik bir değişken olarak kullanan yaklaşımların aksine, EveDroid'de uygulamaların çeşitli davranış modelleri oluşturulur ve bunlar gruplanarak analiz edilir [59]. PlayDrone ve Google Play Store'dan elde edilen çeşitli zararsız uygulama örnekleri ve VirusShare'den elde edilen kötü amaçlı uygulama örneklerinden oluşan veri setleri kullanılarak %99.8 gibi oldukça yüksek bir F1 skoru ölçülmüştür [59].

Literatürdeki en başarılı performans değerlerinin (doğruluk, doğru pozitif / doğru negatif oranları, F1 skoru, vd.) elde edildiği çalışmalar karşılaştırmalı olarak Tablo 1'de özetlenmektedir.

Tablo 1. Literatürdeki en başarılı doğruluk, vd. performans değerlerinin elde edildiği çalışmalar

Araştırmacı Adları / Proje Adı	Yılı	Öznitelikler	Zararlı yazılım örneği (adet)	Zararsız yazılım örneği (adet)	Ulaşılan Doğruluk (Accuracy) Oranı
DroidMat [6]	2012	İzinler, bileşenlerin yerleşimi, haberleşme mesajları ve UPA çağrıları	238	1500	%97.87
MADAM [26]	2012	-	10	50	%93
DroidAPIMiner [8]	2013	İzinler, kritik UPA çağrıları, paket düzeyi bilgi (package-level information) ve uygulama parametreleri	3987	16,000	%99
Peiravian ve Zhu [36]	2013	İzinler ve UPA çağrıları	1260	1250	%96.88
MADS [21]	2013	Karakter katarları	333	333	%94.70
TStructDroid [24]	2013	-	110	110	%90 - %93.6
Yerima ve ark. [35]	2014	UPA tabanlı özellikler, izinler ve Android çerçevesi komutları	2925	3938	%95.8
Elish ve ark. [19]	2015	Veri akışı temelli bir öznitelik	1433	2684	%2.1 (Yanlış Negatif) %2.0 (Yanlış Pozitif)
Yerima ve ark. [28]	2015	Geniş bir öznitelik havuzu	2925	3938	%97.3 - %99
MARVIN [44]	2015	Geniş bir öznitelik seti	15,000	120,000	%98.24
Sayfullina ve ark. [42]	2015	13 farklı özellik grubu	10,000	10,000	%82.10 (Doğru Pozitif Oranı)
Wu ve ark. [23]	2016	Veri akışı UPA'ları	1160	1050	%97.66
Alatwi ve ark. [37]	2016	İzinler, yayın alıcılar ve UPA çağrıları	-	-	%98.72
Coronado-De-Alba ve ark. [51]	2016	İzinler, haberleşme nesneleri, donanım ve yazılım	1531	1531	%97.56
Mlifdect [29]	2017	Statik analizle çıkarılan sekiz farklı öznitelik	3982	4403	%99.7
Alzaylaee ve ark. [31]	2017	Dinamik özellikler	222	1222	%93.1 (Doğru Pozitif Oranı)
Milosevic ve ark. [57]	2017	Kaynak kodu analizi	200	200	%95.1 (F1-skoru)
MAMADROID [38]	2017	Soyutlanmış UPA çağrıları dizisi	35,500	8500	%99 (F1-skoru)
MalDozer [54]	2018	İşlenmemiş UPA çağrıları dizisi	33,000	38,000	%96 - %99 (F1-skoru)
Pektaş ve ark. [45]	2018	Davranışsal özellikler	2000	-	%89
AUNTIEDROID [39]	2018	Dinamik analiz kullanarak çıkarılmış davranışsal modeller	2568	2692	%92 (F1-skoru)
EVEDROID [59]	2019	Sistem eylemlerindeki davranışsal örüntüler	28,848	14,956	%99.8 (F1-skoru)

3. SONUÇLAR (CONCLUSIONS)

Bir önceki bölümde incelenmiş olan literatürdeki çalışmaların artıları ve eksileri, özellikle günlük yaşamda kullanılabilirliği ve ticari açıdan fizibiliteleri bu bölümde değerlendirilmiştir.

Yapılan çalışmaların bazılarında sadece kötücül kodların tespitinin ne düzeyde başarılı olduğu ölçülmüş, zararsız kodların ne düzeyde başarılı olarak belirlendiği göz ardı edilmiştir. Bir başka deyişle, bazı araştırmacılar, sadece doğru pozitif oranlarına odaklanmış, doğru negatif oranları ve buna bağlı olarak yanlış pozitif oranı ve ilgili diğer bazı ölçümler göz ardı edilmiştir. Bu kuramsal çalışmalarda kabul edilebilir olsa da, günlük yaşamda ve ticari amaçlı üretilecek kötücül kod tespiti ürünlerinin kullanıcılar tarafından benimsenmesi açısından bakıldığında yanıltıcı bir yaklaşımdır. Çünkü kullanıcılar ve ticari yazılım firmaları açısından, kullanmakta oldukları bir kötücül yazılım tespiti uygulamasının cihazları zararlı kodlardan ne kadar başarıyla koruyabilmesi yanı sıra, çalıştırmak istedikleri iş veya eğlence amaçlı zararsız uygulamaların da gereksiz yere engellenmemesi, yani, yanlış pozitif alarmla zararsız uygulamaların kullanım dışı kalmaması da çok önemlidir. Bu nedenle, bu gibi araştırma ve yayınlarda, hem pozitif hem negatif sınıf aynı derecede önemle değerlendirilmeli ve Kappa istatistiği, Matthews correlation coefficient, vb. ölçümler de yapılmalıdır.

Literatürdeki birçok çalışmada gözlemlenen ve önem verilen bir başarı ölçümünün de doğruluk (accuracy) olduğu görülmüştür. Eğer her iki sınıfa ait kayıt sayısı eşit ise, başka bir deyişle, pozitif ve negatif sınıflarına ait kayıtların adetleri birbirine çok yakın ya da eşit ise, doğruluk değerine bakılması uygun bir yaklaşımdır. Öte yandan, bu çalışmaların bazılarında her iki sınıfa ait örnek adetleri çok farklıdır yani dengesiz veri (imbalanced data) durumu vardır. Bu gibi dengesizlik durumlarında ise doğruluk değerine bakmak yanıltıcı bir yaklaşımdır. Bunun nedeni ise, doğruluk ölçümü iki ayrı sınıfa ait doğruluk performanslarının ağırlıklı ortalamasını hesaplamakta yani mikro ortalamaya (micro average) bakmaktadır. Zararlı ve zararsız kod örneği adetlerinin birbirinden çok farklı olduğu durumlarda ise, her bir sınıfa ait doğru pozitif oranı hesaplanıp ayrı ayrı irdelenmeli ve genel doğruluk değeri olarak da bu iki sonucun aritmetik ortalaması yani makro ortalaması (macro average) alınmalıdır. Basit bir örnek vermek gerekirse, 10000 adet zararlı kod ve 500 adet zararsız kod örneği içeren bir veri seti için, 9950 adet zararlı kod ve 400 adet zararsız kod başarıyla tespit ediliyor ise, doğruluk değeri yani mikro ortalamasına bakıldığında 0.985 gibi çok yüksek bir değer edilir. Oysa bu yanıltıcıdır çünkü makro ortalama hesaplandığında 0.897 değeri elde edilecektir, bu da göreceli olarak mikro ortalama yani doğruluk değerinden oldukça düşük bir değerdir.

Birçok çalışma ayrı ayrı detaylı olarak incelendiğinde çok önemli ortak bir olgu ortaya çıkmıştır. Kötücül yazılım tespitinde yapay öğrenme yaklaşımlarında yüksek başarı oranlarının yakalanmasında en önemli ve esas etkenin, seçilen algoritma ve o algoritmanın katsayı, parametreleri, vb. olmadığı, öznitelik seçimi / azaltılması (feature selection / reduction) aşaması ve burada kullanılan yöntemlerdir. Kötücül kod ya da zararsız kodların en belirleyici / ayırt edici özelliklerini içeren öznitelikler başarıyla elde edildikten sonra çoğu yapay öğrenme algoritmasının 0.95 ve üzeri tespit oranları ve doğruluk değerlerine ulaşabildiği görülmektedir. Android uygulamalarında ve ilgili dosyalarda çok farklı tipte ve binlerce sayıda öznitelik çıkabileceği düşünülürse, olabildiğince az sayıda ve aynı zamanda en belirleyici, en etkili özniteliklerin belirlenip seçilmesinin çok zor ve aynı zamanda çok kritik bir işlem olduğu anlaşılmaktadır. Klasik istatistiksel öznitelik seçimi yöntemleri yanı sıra, sınıflandırma algoritmalarına bağlı ya da alt küme seçimi, Relief, vb. yöntemler veya metin madenciliği temelli öznitelik seçimleri kullanılarak başarılı sonuçlar elde edilebilmektedir. Dolayısıyla, yapılacak benzer çalışmalarda, seçilecek ya da yeni tasarlanacak bir yapay öğrenme algoritmasından daha öncelikli olarak, öznitelik seçimi ve azaltılması sürecine odaklanılması başarı olasılığını yükselten daha uygun bir strateji olacaktır.

Bazı araştırmacılar, çalışmalarında statik kod analizi ile örnekleri elde edip incelemiş, öte yandan diğer bazı başka araştırmacılar da dinamik kod analizi yaklaşımı ile zararlı kodların çalışma biçimleri ya da eylemlerinden öznitelik çıkartmaya çalışmıştır. Bu araştırmacıların bazıları da, ilgili yayınlarında kendi tercih ettikleri yöntemin daha uygun ya da en geçerli olduğu gibi varsayımlarda bulunmuştur. Bu oldukça tartışmalı ve yanıltıcı olabilecek bir yaklaşımdır. Statik ve dinamik kod analizi yöntemlerinin her birisinin kendine özgü avantaj ve dezavantajları bulunmaktadır. Örneğin, statik kaynak kod analizine bağlı öznitelik seçimi durumlarda, gizleme (obfuscation) özelliği olan kötücül yazılımların tespiti olanaksız hale gelmektedir. Diğer yandan, dinamik kod analizi ve zararlı kodların çalışma tarzlarına bakarak öznitelik elde edilmesi çoğunlukla çok zaman alıcı, zahmetli olması yanı sıra, bazı zararlı kodlar yapısı gereği gözlemlenebilecek bir eylem ya da tarz sergilememekte ya da zararsız uygulamalarla tamamen aynı biçimde eylemler sergileyerek aldatıcı olabilmektedir. Bu nedenle, statik ya da dinamik kod analizinden birisini göz ardı etmek ya da bunlardan hangisinin daha başarılı olacağını tartışmaktansa, bir önceki paragrafta değinildiği şekilde,

hangi özniteliklerin seçileceği ve kullanılacağına öncelikle odaklanılmalıdır.

Bilgi güvenliği yönetimi insan, süreç ve teknolojinin bileşkesidir ve bu nedenle mobil sistemlerin ve mobil uygulamaların güvenliğinde de insan unsurunun ayrıca ele alınması gerekmektedir. Bilgisiz veya bilinçsiz şekilde Android uygulamalarının kullanılması durumunda, en gelişmiş güvenlik teknolojileri bile yetersiz kalabilir. Anti-virüs, vb. teknolojik önlemlere rağmen kullanıcıların dikkatsizliği, hatalı ve bilinçsiz kullanımları nedeni ile ilgili bilgi güvenliği risklerinin gerçekleşme olasılığı ve etkisi yüksek düzeyde olabilir. Son kullanıcılarda bilgi güvenliği farkındalığı oluşturacak çeşitli eğitimler, sosyal sorumluluk projeleri, kampanyaların yapılması sağlanmalıdır. Zararlı kodlarla mücadelede başarılı olmak için sadece teknolojiye odaklanılmamalı, insan unsuru da sürece mutlaka katılmalıdır.

Çalışmaların bazılarında en başarılı sonuçların alındığı yapay öğrenme algoritmalarının aslında gerçek anlamda eğitilmiş bir model olmayıp kara kutu (black box) mimariler denilen en son eğitilmiş ağırlıklar ve hiper-parametrelere bağlı olduğu (yapay sinir ağları, derin öğrenme, DVM, vb.) görülmektedir. Bu durumun, akademik bakış açısından herhangi bir sakıncası olmasa da, eğer uzun vadede ticari bir ürün haline getirilecek ve endüstriyel süreçlere katılacak standart bir model elde edilmesi gerekiyor ise sıkıntılı durumlar yaratacaktır. Karar ağaçları, kural tabanlı sınıflandırıcılar ve bu tür algoritmaların kolektif öğrenme yaklaşımları gerçek dünyadaki ticari uygulamalar için çok daha uygundur çünkü bu tip algoritmaların eğitimi sonunda somut bir model oluşmaktadır. Bu konuya bağlı olarak, bazı çalışmaların konuya tamamen kuramsal ve akademik bakış açısıyla yaklaşarak olabilecek en başarılı tespit değerlerini elde etmeye odaklanıldığı görülmüştür. Çok yavaş çalışan ya da eğitim süresi aşırı uzun süren, yüksek bellek gereksinim duyan, eğitim sonunda bir model oluşturmayan algoritmaların günlük yaşamdaki uygulamalarda, endüstriyel süreçlerde ve pratikte kullanımı pek olası olamamaktadır. Doğru pozitif, doğruluk, vb. değerleri göreceli olarak daha düşük olsa da, az bellek gerektiren, çok hızlı çalışan, eğitim süresi çok kısa süren, modellenebilen yaklaşımların günlük yaşamda ve ticari uygulamalarda şansı daha fazla olacaktır.

Android veya diğer teknolojik platformlarda kötücül kod tespit eden anti-virüs, kişisel güvenlik duvarı, vb. yazılımların tasarlanırken, test edilirken ve piyasaya sürülürken tek kıstas doğruluk, vb. ölçümlerde mükemmele yakın sonuç elde edilmesi olmamalıdır. Üretim maliyeti, çalışma hızı, kullanım kolaylığı, donanım gereksinimi, kaynak tüketimi, yaygınlaşabilmesi, diğer uygulamalarda ya da cihazın kendisinde beklenmedik sorunlara yol açmaması gibi kıstaslar da kurumlar ve bireyler için oldukça önemlidir. Bu nedenle, bu alanda yapılacak çalışmalar ve makalelerde, ortaya konan yeni yöntemlerin ve çözümlerin bu açılardan da irdelenmesinin daha gerçekçi ve etkili bir yaklaşım olacağı düşünülmektedir.

KAYNAKLAR (REFERENCES)

[1] A.D. Schmidt, H. G. Schmidt, J. Clausen, K.A. Yuksel, O. Kiraz, A. Camtepe, S. Albayrak, "Enhancing security of Linux-based Android devices", **15th International Linux Conference**, Hamburg, Germany, 2008.

[2] A.D. Schmidt, R. Bye, H.-G Schmidt, J. Clausen, O. Kiraz, K. A. Yuksel, S. A. Camtepe, S. Albayrak, "Static Analysis of Executables for Collaborative Malware Detection on Android", **(ICC'09) IEEE International Conference on Communications,** Dresden, Germany, 631–635, June 14-18, 2009.

[3] W. Enck, M. Ongtang, P. D. McDaniel, "Understanding Android Security", *IEEE Security & Privacy*, 7(1), 50–57, 2009.

[4] A. P. Fuchs, A. Chaudhuri, J. S. Foster, **SCanDroid: Automated Security Certification of Android Applications**, Technical Report CS-TR-4991, Department of Computer Science, University of Maryland, College Park, November 2009.

[5] T. Bläsing, L. Batyuk, A.-D Schmidt, S. A. Camtepe, S. Albayrak, "An Android application sandbox system for suspicious software detection", **5th International Conference on Malicious and Unwanted Software**, Nancy, Lorraine, France, 55–62, October 19-20, 2010.

[6] D. Wu, C. Mao, T. Wei, H. Lee, K. Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing", **(Asia JCIS), Seventh Asia Joint Conference on Information Security**, Tokyo, Japan, 62–69, August 9-10, 2012.

[7] Internet: Contagio Mobile, http://contagiominidump.blogspot.com, 28.01.2018.

[8] Y. Aafer, W. Du, H. Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android", *Secure Communications, Lect. Notes of the Inst. for CS, Social Informatics and Telecommunications Engineering*, 127, 86–103, 2013.

[9] J. R. Quinlan, "Induction of Decision Tree", *Machine Learning*, 1, 81-106, 1986.

[10] J. R. Quinlan, **C4.5: Programs for Machine Learning**, Morgan Kaufmann, 1993.

[11] D. W. Aha, D. Aha, M. K. Albert, "Instance-Based learning algorithms", *Machine Learning*, 6, 37-66, 1991.

[12] C. Cortes, V. Vapnik, "Support-Vector Networks", *Machine Learning*, 20, 273-297, 1995.

[13] W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. McDaniel, A. N. Sheth, "TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones", **9th USENIX Conference on Operating Systems Design and Implementation**, Vancouver, BC, Canada, 1–6, October 4-6, 2010.

[14] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, I. Molloy, "Using probabilistic generative models for ranking risks of android apps", **(CCS 2012) ACM Conference on Computer and Communications Security,** New York, USA, 241–252, October 16-18, 2012.

[15] P. Lantz, A. Desnos, K. Yang. "Droidbox: Android application sandbox", https://github.com/pjlantz/droidbox.

[16] E. Chin, A. Felt, K. Greenwood, D. Wagner, "Analyzing interapplication communication in Android", **9th International Conference on Mobile Systems, Applications, and Service**s, New York, USA, June 28-July 01, 2011.

[17] I. Burguera, U. Zurutuza, S. Nadjm-Tehrani, "Crowdroid: Behavior based malware detection system for Android", **1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices**, New York, USA, 15–26, October 17-21, 2011.

[18] S. Y. Yerima, S. Sezer, G. McWilliams, I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification", **(AINA 2013) 27th International Conference on Advanced Information Networking and Applications**, Barcelona, Spain, 121–128, March 25-28, 2013.

[19] K. O. Elish, X. Shu, D. D. Yao, B. G. Ryder, X. Jiang, "Profiling user-trigger dependence for Android malware detection", *Computers & Security,* 49, 255–273, 2015.

[20] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, K. Rieck, C. Siemens, "Drebin: Effective and explainable detection of Android malware in your pocket", **(NDSS) 21st Annual Symposium on Network and Distributed System Security**, San Diego, California, February 23-26, 2014.

[21] B. Sanz, I. Santos, J. Nieves, C. Laorden, I. Alonso-Gonzalez, P. G. Bringas, "MADS: Malicious Android Applications Detection through String Analysis", *Network and System Security*, 7873, Lecture Notes in Computer Science, Javier Lopez, Xinyi Huang, Ravi Sandhu (Eds.), 178–191, Springer Berlin Heidelberg, 2013.

[22] E. N. Çinicioğlu, M. Atalay, H. Yorulmaz, "Trafik Kazaları Analizi için Bayes Ağları Modeli", *Bilişim Teknolojileri Dergisi*, 6(2), 41-52, 2013.

[23] S. Wu, P. Wang, X. Li, Y. Zhang, "Effective detection of Android malware based on the usage of data flow APIs and machine learning", *Information and Software Technology*, 75, 17-25, 2016.

[24] F. Shahzad, M. Akbar, S. Khan, M. Farooq, **Tstructdroid: Realtime malware detection using in-execution dynamic analysis of kernel process control blocks on Android**, Technical Report, National University of Computer & Emerging Sciences, Islamabad, Pakistan, 2013.

[25] L. Yan, H. Yin, "DroidScope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis", **21st USENIX Security Symposium**, Bellevue, WA, USA, 569-584, August 8-10, 2012.

[26] G. Dini, F. Martinelli, A. Saracino, D. Sgandurra, "Madam: A multi-level anomaly detector for Android malware", **(MMM-ACNS'12) 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security**, St. Petersburg, Russia, 240–253, October 17-19, 2012.

[27] C. Zheng, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han, W. Zou, "Smartdroid: an automatic system for revealing ui-based trigger conditions in Android applications", **2nd ACM workshop on Security and privacy in smartphones and mobile devices**, Raleigh, NC, USA, 93–104, October 16-18, 2012.

[28] S. Y. Yerima, S. Sezer, I. Muttik, "High accuracy Android malware detection using ensemble learning", *IET Information Security*, 9(6), 313–320, 2015.

[29] X. Wang, D. Zhang, X. Su, W. Li, "Mlifdect: Android malware detection based on parallel machine learning and information fusion", *Security and Communication Networks*, 1-14, 2017.

[30] Y. Zhou, X. Jiang, "Dissecting Android Malware: Characterization and Evolution", **33rd IEEE Symposium on Security and Privacy**, San Francisco, CA, USA, 95–109, May 20-23, 2012.

[31] M. K. Alzaylaee, S. Y. Yerima, S. Sezer, "EMULATOR vs REAL PHONE: Android malware detection using machine learning", **3rd ACM International Workshop on Security and Privacy Analytics**, Scottsdale, Arizona, USA, March 22 - 24, 2017.

[32] Internet: SantokuLinux, https://santoku-linux.com, 07.11.2018.

[33] B. Amos, H. Turner, J. White, "Applying machine learning classifiers to dynamic Android malware detection at scale", **(IWCMC 2013) 9th International Wireless Communications and Mobile Computing Conference**, Cagliari, Sardinia, Italy, July 1-5, 2013.

[34] W-C. Wu, S-H. Hung, "DroidDolphin: A dynamic Android malware detection framework using big data and machine learning", **International Conference on Research in Adaptive and Convergent Systems**, Towson, MD, USA, October 05-08, 2014.

[35] S. Y. Yerima, S. Sezer, I. Muttik, "Android malware detection using parallel machine learning classifiers", **8th International Conference on Next Generation Mobile Apps, Services and Technologies**, Oxford, UK, 37-42, September 10-12, 2014.

[36] N. Peiravian, X. Zhu, "Machine learning for Android malware detection using permission and API calls", **IEEE 25th International Conference on Tools with Artificial Intelligence,** Herndon, VA, USA, November 4-6, 2013

[37] H. A. Alatwi, T. Oh, E. Fokoue, B. Stackpole, "Android malware detection using category-based machine learning classifiers", **17th Annual Conference on Information Technology Education**, Boston, MA, USA, September 28 - October 01, 2016.

[38] E. Mariconti, L. Onwuzurike, P. Andriotis, E. D. Cristofaro, G. Ross, G. Stringhini, "MaMaDroid: Detecting Android malware by building markov chains of behavioral models", *arXiv:1612.04433*, https://arxiv.org/abs/1612.04433, 2017.

[39] L. Onwuzurike, M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, E. D. Cristofaro, "A family of droids: Analyzing behavioral model based Android malware detection via static and dynamic analysis", *arXiv:1803.03448*, https://arxiv.org/abs/1803.03448, 2018.

[40] M. Almeida, M. Bilal, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Varvello, J. Blackburn, "CHIMP: Crowdsourcing Human Inputs for Mobile Phones", **(WWW 2018) World Wide Web Conference**, Lyon, France, April 23 – 27, 2018.

[41] Internet: VirusShare, https://virusshare.com, 27.08.2018.

[42] L. Sayfullina, E. Eirola, D. Komashinsky, P. Palumbo, Y. Miche, A. Lendasse, J. Karhunen, "Efficient detection of zero-day Android malware using normalized Bernoulli Naive Bayes", **IEEE Trustcom/BigDataSE/ISPA,** Helsinki, Finland, August 20-22, 2015.

[43] R. Dhaya, M. Poongodi, "Detecting software vulnerabilities in Android using static analysis", **IEEE International Conference on Advanced Communications, Control and Computing Technologies**, Ramanathapuram, India, May 8-10, 2014.

[44] M. Lindorfer, M. Neugschwandtner, C. Platzer, "MARVIN: Efficient and comprehensive mobile app classification through static and dynamic analysis", **IEEE 39th Annual Computer Software and Applications Conference,** Taichung, Taiwan, July 1-5, 2015.

[45] A. Pektaş, M. Çavdar, T. Acarman, "Android malware classification by applying online machine learning", **(ISCIS 2016) International Symposium on Computer and Information Sciences,** Kraków, Poland, 72-80, October 27–28, 2016.

[46] Internet: CuckooSandbox, https://cuckoosandbox.org, 05.07.2018.

[47] Y-W. Chen, C-J. Lin, "Combining SVMs with various feature selection strategies", *Feature Extraction,* 207, Studies in Fuzziness and Soft Computing, I. Guyon, M. Nikravesh, S. Gunn, L. A. Zadeh (Eds.), 315-324. Berlin, Heidelberg, 2016.

[48] M. Damshenas, A. Dehghantanha, K-K. R. Choo, R. Mahmud, "M0Droid: An Android behavioral-based malware detection model", *Journal of Information Privacy and Security*, 11(3), 141-157, 2015.

[49] Internet: OWASP Seraphimdroid, https://github.com/nikolamilosevic86/owasp-seraphimdroid, 15.08.2018.

[50] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, J. Blasco, "Dendroid: A text mining approach to analyzing and classifying code structures in Android malware families", *Expert Systems with Applications,* 41(4), 1104-1117, 2014.

[51] L. D. Coronado-De-Alba, A. Rodríguez-Mota, P. J. E. Ambrosio, "Feature selection and ensemble of classifiers for Android malware detection", **8th IEEE Latin-American Conference on Communications (LATINCOM),** Medellin, Colombia, November 15-17, 2016.

[52] G. Suarez-Tangil, G. Stringhini, "Eight years of rider measurement in the Android malware ecosystem: Evolution and lessons learned", *arXiv:1801.08115*, https://arxiv.org/abs/1801.08115, 2018.

[53] K. Allix, T F. Bissyandé, J. Klein, Y. L. Traon, "AndroZoo: Collecting millions of Android apps for the research community", **IEEE/ACM 13th Working Conference on Mining Software Repositories**, Austin, TX, USA, May 14-22, 2016.

[54] E. B. Karbab, M. Debbabi, A. Derhab, D. Mouheb, "MalDozer: Automatic framework for Android malware detection using deep learning", *Digital Investigation*, 24, 48-59, 2018.

[55] M. A. Kızrak, B. Bolat, "Derin Öğrenme ile Kalabalık Analizi Üzerine Detaylı Bir Araştırma", *Bilişim Teknolojileri Dergisi*, 11(3), 263-286, 2018.

[56] E. Dandil, K. K. Çevik, "Yapay Sinir Ağları İçin .NET Platformunda Görsel Bir Eğitim Yazılımının Geliştirilmesi", *Bilişim Teknolojileri Dergisi*, 5(1), 19-28, 2012.

[57] N. Milosevic, A. Dehghantanha, K-K. R. Choo. "Machine learning aided Android malware classification", *Computers & Electrical Engineering*, 61, 266-274, 2017.

[58] L. Li, D. Li, T F. Bissyande ve ark., "On locating malicious code in piggybacked Android apps", *Journal of Computer Science and Technology*, 32(6), 1108–1124, 2017.

[59] T. Lei, Z. Qin, Z. Wang, Q. Li, D. Ye, "EveDroid: Event-Aware Android Malware Detection Against Model Degrading for IoT Devices", *IEEE Internet of Things Journal*, 6(4), 6668 - 6680, 2019.