Bilge International Journal of Science and Technology Research
Web : http://dergipark.gov.tr/bilgesci - **E-mail:** kutbilgescience@gmail.com

Received: 25.02.2019
Accepted: 25.03.2019
DOI: 10.30516/bilgesci.531801

ISSN: 2651-401X
e-ISSN: 2651-4028
3 (1), 67-76, 2019

Kötü Amaçlı Windows Çalıştırılabilir Dosyalarının Derin Öğrenme ile Tespiti

Mahmut Tokmak[1*], Ecir Uğur Küçüksille[2]

Özet: Günümüz internet çağında kötü amaçlı yazılımlar, bilgi güvenliği açısından ciddi ve gelişen bir tehdit olarak karşımıza çıkmaktadır. Bu nedenle kötü amaçlı yazılımların tespit edilmesi, kötü amaçlı yazılımın yol açabileceği zararların önlenmesi açısından son derece önem arz etmektedir. Bu çalışmada Windows uygulama programlama arayüzü (API) çağrıları ve Windows çalıştırılabilir dosyalarının opsiyonel başlık bölümünün ihtiva ettiği alanlar analiz edilerek kötü amaçlı yazılımlar tespit edilmeye çalışılmıştır. Çalışmada, kötü amaçlı ve kötü amaçlı olmayan çalıştırılabilir dosyalarından oluşan bir veri seti oluşturulmuştur. Veri setinde, 592 kötü amaçlı olmayan yazılım ve 283 kötü amaçlı yazılım olmak üzere 875 Windows çalıştırılabilir dosyası kullanılmıştır. Veri setindeki her bir çalıştırılabilir dosya, Windows uygulama programlama arayüzü çağrıları ve opsiyonel başlık alanları ele alınarak vektörel olarak ifade edilmiştir. Öznitelik vektörü üzerinde temel bileşen analizi yapılarak boyut indirgeme işlemi yapılmıştır. İndirgenen öznitelikler Derin Öğrenme ile eğitilip test edilerek kötü amaçlı yazılım tespiti gerçekleştirilmiştir. Çalışmanın sonunda Derin Öğrenme ile % 100 doğruluk değerine erişilmiştir.

Keywords: Kötü amaçlı yazılım tespiti, Bilgi ve bilgisayar güvenliği, Derin öğrenme.

Detection of Windows Executable Malware Files with Deep Learning

Abstract: In today's internet age, malware emerges as a serious and growing threat in terms of information security. Therefore, detecting malware is extremely important in terms of preventing harm that malware may cause. In this study, by analyzing Windows Application Programming Interface (API) calls and the optional header sections of Windows executable files, it was tried to detect malware. A data set consisting of malware and benign executable files was created. In this study, 875 portable executable files were used, 283 of them are benign and 592 of them are malware. Each portable executable file in the data set is expressed in vectors by the taking into account Windows application programming interface calls and the optional header sections. Dimension reduction was made on feature vector. The reduced attributes were trained and tested by Deep Learning and detecting malware was achieved. At the end of the study, it was achieved 100% accuracy with Deep Learning.

Keywords: Malware detection, Information and computer security, Deep learning.

[1]Isparta Uygulamalı Bilimler Üniversitesi, Gelendost Meslek Yüksekokulu, Maliye Bölümü, Gelendost/Isparta-Türkiye
[2]Süleyman Demirel Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Isparta-Türkiye
*Corresponding author (İletişim yazarı): mahmuttokmak@sdu.edu.tr

Citation (Atıf): Tokmak, M., Küçüksille, E.U. (2019). Kötü Amaçlı Windows Çalıştırılabilir Dosyalarının Derin Öğrenme İle Tespiti. Bilge International Journal of Science and Technology Research, 3(1): 67-76.

1. Giriş

Bilgisayar teknolojileri her geçen gün daha da gelişmekte ve yaygınlaşmaktadır. Kişisel ve kurumsal kullanımda yapılan işlemler elektronik platformlara aktarılarak kolaylık sağlanmaktadır. Elektronik ortamlara aktarılan bilgilerin güvenliğini sağlamak ise başlı başına bir önem derecesine sahip ve çözülmesi gereken bir problem olarak karşımıza çıkmaktadır. Çünkü kötü amaçlı yazılımların yarattığı tehditler sayısal olarak ve çeşitlilik olarak artmaktadır. Hatta tespit edilen kötü amaçlı bir yazılım şekil değişikliğine uğratılıp tekrar daha güçlü olarak karşımıza çıkabilmektedir. Bu tehditlerin başında kötü amaçlı yazılım (malware) ve casus yazılım (spyware) gelmektedir.

Kötü amaçlı yazılım (malware); İngilizce "malicious software" kelimelerinin kısaltılması ile adlandırılmıştır. Girmiş olduğu sistemde zarar vermeyi amaçlayan bu yazılımlar bilgisayar virüsleri, solucanlar (worms), arka kapılar (backdoor), Truva atları (Trojan horse) ve casus yazılımlar(spyware) olarak çeşitlilik göstermektedir. Kötü amaçlı yazılımlar, hassas verilerin çalınması, şifrelenmesi, silinmesi, temel fonksiyonların değiştirilmesi ve kullanıcıların izni olmadan bilgisayar faaliyetlerinin izlenmesi de dahil olmak üzere çeşitli işlevler gerçekleştirebilirler (Ye vd., 2008; Shabtai vd., 2009).

Kötü amaçlı yazılım türleri, belirli bir amaç için tasarlanmış işlevleri veya özellikleri içerir. Örneğin fidye yazılımları (ransomware), bir kullanıcının sistemine bulaşarak verileri şifrelemek için tasarlanmıştır. Siber suçlular, sistem verilerinin şifresini çözmek karşılığında kurbandan fidye talep etmektedirler. Kök kullanıcı takımı (rootkit), mağdurun sistemine yönetici seviyesinde erişim sağlamak için tasarlanmış kötü amaçlı yazılımlardır. Yazılım kurulduktan sonra, tehdit aktörlerine, kök veya sisteme ayrıcalıklı erişim verirler. Arka kapı virüsü veya uzaktan erişimli Trojan (RAT), sistemde gizlice bir arka kapı oluşturur. Kullanıcıya veya sistemin güvenlik programlarına fark ettirmeden tehdit aktörlerinin uzaktan erişmesine izin verirler (Ye vd., 2008; Bazrafshan vd., 2013; Basu vd., 2016).

Son zamanlarda, kötü amaçlı yazılımların aşırı bir şekilde yaygınlaştığı, nicel ve nitel anlamda arttığı görülmektedir. McAfee Labs'ın 2018 yılının mart ayında yayınladığı rapora göre; 63,4 milyon yeni kötü amaçlı yazılım örneği ile tüm zamanların en yüksek rakamına eriştiği belirtilmiştir (McAfee, 2018). Her gün binlerce sistem bu kötü amaçlı yazılımların verdiği zararlara maruz kalmaktadır. Dünyada en çok kullanılan Windows işletim sistemi de kötü amaçlı yazılımların hedefi halindedir. Microsoft Windows işletim sistemlerinin dünyadaki kullanım oranı 2018 Mayıs ayı itibariyle %76 olarak tespit edilmiştir (W3schools, 2018). Yüksek oranda kullanım oranına sahip olan Windows işletim sistemini hedef alan kötü amaçlı yazılımlar exe, com, dll gibi dosya uzantısına sahip olabilmektedirler (Belaoued ve Mazouzi, 2016). Böylesine yaygın kullanım oranına sahip olması ve gün geçtikçe artan kötü amaçlı yazılımların hedefi olması nedeniyle; Windows işletim sistemi ve çalıştırılabilir dosya formatı PE (Portable Executable) tipindeki kötü amaçlı yazılımlar çalışmada odak konusu olmuştur.

Kötü amaçlı yazılımların tespiti için üç tür yaklaşım yoğun olarak kullanılmakta ve öne çıkmaktadır. Bunlar; imza tabanlı tespit tekniği, davranış tabanlı tespit tekniği ve sezgisel tespit tekniğidir (Bazrafshan vd., 2013; Belaoued ve Mazouzi, 2016; Darshan ve Jaidhar, 2018).

İmza tabanlı tespit tekniği, birçok antivirüs yazılımının kullandığı bir tekniktir. İmza ise tek olan ve kısa bir byte dizisinden oluşmaktadır. Bu teknik; bilinen ve daha önceden tespit edilmiş bir kötü amaçlı yazılımın imzası çıkarılıp veri tabanına kaydedildikten sonra test edilecek yazılımın imzasıyla karşılaştırma esasına göre çalışmaktadır. Ancak bilinmeyen, karşılaşılmamış yazılımlarda başarısız olmaktadır. Kötü amaçlı yazılımın zararlarının ortaya çıkması, tespit edilmesi ve veritabanı güncellemesi sürecinde geçen zamanda kötü amaçlı yazılım zarar vermeye devam etmektedir. Polimorfik ve metaformik kötü amaçlı yazılımları tespit edememesi bu yöntemin bir dezavantajı olarak karşımıza çıkmaktadır (Ye vd., 2008; Belaoued ve Mazouzi, 2016; Lim, 2016; Darshan ve Jaidhar, 2018; Kumar vd., 2017).

Davranış tabanlı tespit tekniği, yazılımın çalışma anındaki davranışlarını izleyerek tespit etmeye çalışmaktadır. Kötü amaçlı yazılımın, mevcut sistemden izole edilmiş kum havuzu (sandbox) yada bir sanal makine (virtual machine) üzerinde çalıştırılarak API çağrıları, sistem çağrıları, internet trafiği gibi özelliklerin toplanıp izlenmesi ilkesine dayalı bir tespit tekniğidir (Bazrafshan vd., 2013; Belaoued ve Mazouzi, 2014; Belaoued ve Mazouzi, 2016; Zatloukal ve Znoj, 2017; Darshan ve Jaidhar, 2018). Bu yöntem birçok çalışmada dinamik analiz

olarak da adlandırılmaktadır (Bazrafshan vd., 2013; Basu vd., 2016; Belaoued ve Mazouzi, 2016; Zatloukal ve Znoj, 2017). Ancak kötü amaçlı yazılımın davranışlarını sergileyen özelliklerin toplanması aşaması statik olarak yapılabilmektedir (Bazrafshan vd., 2013). Statik analiz yaklaşımı, analiz edilecek yazılımı çalıştırmadan yazılım kodları hakkında detaylı bilgi elde etmeyi amaçlayan yaklaşımdır. Statik analiz yaklaşımında yazılım sisteme kurulmadığı için incelemenin yapıldığı sistem kötü amaçlı yazılımın doğurabileceği zararlardan etkilenmemektedir (Salehi vd., 2014; Kabakuş vd., 2015; Zatloukal ve Znoj, 2017). Davranış tabanlı tespit tekniğinin, imza tabanlı tekniğin tespit edemediği, bilinmeyen ve polimorfik kötü amaçlı yazılımları tespit edebilmesi bir avantaj olmakla birlikte tarama zamanının uzun olması, metamorfik kötü amaçlı yazılım türlerini tespit edememesi ve Yanlış Pozitif Oranı (False Positive Rate, kötü amaçlı olmayıp kötü amaçlı olarak etiketlenen yazılım oranı) istenen düzeyde olmaması bir dezavantaj olarak karşımıza çıkmaktadır (Bazrafshan vd., 2013; Belaoued ve Mazouzi, 2016; Darshan ve Jaidhar, 2018).

Bahsedildiği gibi, imza tabanlı ve davranış tabanlı kötü amaçlı yazılım tespit yöntemlerinin bazı dezavantajları bulunmaktadır. Bu dezavantajların üstesinden gelmek için sezgisel tespit teknikleri geliştirilmiştir. Sezgisel tespit teknikleri, yürütülebilir bir dosyanın davranışını öğrenmek için veri madenciliği ve makine öğrenme tekniklerini kullanmaktadırlar. Bu konudaki çalışmaları Schultz vd. Naive Bayes ve Multi Naive Bayes yöntemlerini kullanarak başlatmışlardır (Schultz vd., 2001; Bazrafshan vd., 2013). Sezgisel tabanlı tespit sistemleri, kötü amaçlı yazılımları tespit etmek için, sistem çağrıları, API çağrıları, Opkodları ve başlık bilgisi gibi yapısal bilgileri incelenmektedir ki bu bilgiler makine öğrenme ve veri madenciliği yöntemlerinde özellik vektörü olarak kullanılmaktadır. Sezgisel tabanlı tespit tekniği, bilinmeyen ve metamorfik kötü amaçlı yazılımları tespit edebilmektedir. Ancak Yanlış Pozitif Oranı halen sorun olarak araştırmacıların karşısına çıkmaktadır. Bu sorunu aşmak için çalışmalarda, yazılımlara ait farklı özellikler, farklı makine öğrenme yöntemleri ve veri madenciliği teknikleri kullanılmaktadır (Schultz vd., 2001; Ye vd., 2008; Ye vd., 2010; Bazrafshan vd., 2013; Belaoued ve Mazouzi, 2016; Darshan ve Jaidhar, 2018).

Bu çalışmada, sezgisel tespit tabanlı PE çalıştırılabilir dosya türleri için kötü amaçlı yazılım tespit sistemi geliştirilmiştir. Geliştirilen sistem, öznitelik çıkarma, boyut indirgeme ve karar verme aşamalarını içermektedir. Öznitelik olarak PE dosyalarının API çağrıları ve PE Opsiyonel Başlığı (IOH: Image Optional Header) özellikleri kullanılmıştır. Öznitelik sayısı fazla olduğundan Temel Bileşen Analizi (PCA: Principal Component Analysis) ile boyut indirgeme işlemi yapılmıştır. Karar verme aşamasında ise Derin Öğrenme (DL: Deep Learning) yöntemi kullanılmıştır. Tüm aşamalarda Python programlama dili kullanılmıştır.

Bu konuda literatürde yapılmış çeşitli çalışmalara bakıldığında; Schultz vd. (2001); PE dosyalarındaki metinleri, API çağrılarını bayt dizisi özelliklerini araştırarak kötü amaçlı yazılımları tespit etmişlerdir. Sınıflandırıcı olarak Naive Bayes ve Multi Naive Bayes yöntemlerini kullanmışlardır. Önerdikleri yöntemleri imza tabanlı tespit tekniği ile karşılaştırmışlardır. Naive Bayes yöntemi ile % 97.11, Multi Naive Bayes yöntemi ile % 96.88 oranında doğruluk elde etmişlerdir.

Ye vd. (2008); IMDS (Intelligent Malware Detection System) , Akıllı Kötü Amaçlı Tespit Sistemi, adlı bir tespit sistemi önermişlerdir. PE dosyalarındaki API çağrılarını analiz etmişlerdir. Nesne yönelimli birliktelik tabanlı FP-Growth algoritmasını kullanarak sınıflandırma yapmışlardır. Önerdikleri sistemin tespit oranını % 97,18, doğruluk oranını % 93,07 olarak belirtmişlerdir.

Wang vd. (2009); Windows ortamlarında API dizilerinin sergilemiş olduğu şüpheli davranışların analizine dayalı bir kötü amaçlı yazılım tespit etme yaklaşımı sunmuşlardır. Şüpheli davranışları saptamak ve virüsleri tespit etmek için Bayes algoritmasını uygulamışlar ve % 93,71 doğruluk oranı elde etmişlerdir.

Ye vd. (2010); IMDS adlı daha önceki kötü amaçlı yazılım tespit sisteminin iyileştirilmesi olarak CIMDS' yi önermişlerdir. Daha önceki çalışmaları gibi nesne yönelimli birliktelik madenciliği kullanmışlardır. Kural seçme, kural budama ve kural sıralama işlemleri için Ki-kare metodunu kullanmışlardır. Önerdikleri sistemde tespit oranını % 89,52, doğruluk oranını %71,35 olarak belirtmişlerdir.

Byrd vd. (2014); öğrenme tabanlı sınıflandırıcılarla kötü amaçlı yazılım tespit etmeyi amaçlamışlardır.

Çalışmalarında üç öğrenme tabanlı sınıflandırıcı kullanmışlardır. Elde ettikleri bir veri kümesindeki kötü amaçlı kodların analizi ve tespiti için; Fisher doğrusal sınıflandırıcı, otomatik sinir ağı sınıflandırıcısı ve destek vektör makinelerini kullanmışlardır. Veri setini oluşturmak için genel sandbox teknikleri ve API Monitor v2 programını kullanmışlardır. API Monitor v2 yazılımını kullanarak API çağrılarını çıkarmışlardır Önerdikleri tespit mekanizmasında, Fisher doğrusal sınıflandırıcı % 83,2, otomatik sinir ağı sınıflandırıcısı % 85,7, destek vektör makineleri kullanarak yaptıkları sınıflandırma % 94,9 doğruluk oranına ulaşmıştır.

Lim (2016); API işlev çağrılarının bilgilerini analiz ederek yazılımın kötü amaçlı davranışlarını tespit etme yaklaşımını önermiştir. Tespit mekanizmasında dinamik API işlev çağrılarını analiz ederek, kötü niyetli yazılım davranışlarını saptamaya çalışmıştır. Analizin verimliliğini ve toleransını artırmak için, kötü amaçlı davranışları k-gram setleri olarak ayarlanmış ve k-gram kümeleri ve API işlev çağrısı arasındaki benzerliği hesaplayarak tanımlama yapmıştır.

Gupta vd. (2016); kötü amaçlı yazılımları tespit etmek için Windows API çağrı dizileri kullanmışlardır. Windows API çağrı dizilerini yakalamak için Microsoft'un Detours kütüphanesini kullanmışlardır. Toplam 534 API setini işlevlerine göre 26 kategoriye ayırmışlar ve bu kategorilere göre kötü amaçlı yazılımları tespit etmişlerdir. API çağrı modelini çıkarmak için n-gram analizi yapmışlardır.

Hardy vd. (2016); DL4MD adını verdikleri çalışmada, Stacked AutoEncoders (SAE) modelini kullanan DL mimarisi ile kötü amaçlı yazılım tespiti önermişlerdir. PE dosyalarından API çağrıları çıkarılarak elde ettikleri verileri, DL girdisi olarak kullanmışlardır. Önermiş oldukları yöntemin doğruluk oranını % 96,85 olarak belirtmişlerdir.

Belaouded ve Mazouzi (2016); API çağrıları ve PE Opsiyonel Başlık özelliklerini kullanarak Ki kare özellik seçimine dayalı bir sistem önermişlerdir. Sınıflandırma işlemlerini veri madenciliği yazılımı WEKA kullanarak yapmışlardır. Karar ağacı (J48), AdaBoostM1 algoritması ile Hızlandırılmış Karar Ağacı (Boosted Decision Tree), Döndürme Ağaç yapısı (Rotation Forest), Rastgele Orman (Random Forest) sınıflandırma algoritmalarının kötü amaçlı yazılımı tespit etme performanslarını değerlendirmişlerdir. En yüksek doğruluk oranını Hızlandırılmış Karar Ağacı yöntemi ile % 98,17 olarak belirtmişlerdir.

Kolosnjaji vd. (2016); birçok çalışmada olduğu gibi API çağrılarını kullanarak DL ile tespit sistemi önermişlerdir. Çalışmalarında DL yönteminin performansını Saklı Markov Modeli (Hidden Markov Model) ve Destek Vektör Makineleri (Support Vector Machine) ile karşılaştırmışlardır. DL yöntemini diğer modellere karşı daha başarılı olarak tespit etmişler ve % 89,4 doğruluk oranı belirtmişlerdir.

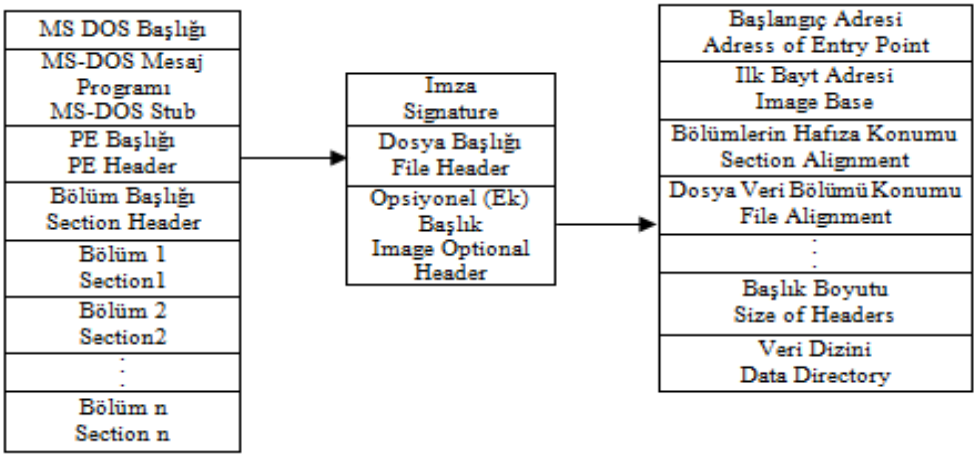
Darshan ve Jaidhar (2018); Filtre Tabanlı Öznitelik Seçim yöntemlerinin, PE kötü amaçlı dosyalarının sınıflandırılmasındaki performanslarını araştırmışlardır. Kullandıkları yöntemler; Ayırt Edici Özellik Seçici yöntemi, Karşılıklı Bilgi yöntemi, Kategorik Orantısal Fark yöntemi, ve Darmstadt İndeksleme Yaklaşımıdır. Öznitelik olarak PE dosya formatının Opsiyonel Başlık verilerini kullanmışlardır.

Cui vd. (2018); DL metodu ile tespit yöntemi önermişlerdir. İkili dosya (binary file) formatındaki dosyaları gri tonlamalı görüntü formatına çevirmişler ve sisteme girdi olarak kullanmışlardır. Çalışmalarında % 94,5 doğruluk oranına ulaşmışlardır.

2. Materyal ve Yöntem

Önerilen yöntemde Windows 10 işletim sisteminde kullanılan taşınabilir, çalıştırılabilir (PE) formattaki dosyalar kullanılmıştır. PE dosyaları, Şekil 1'de verildiği üzere, MS-DOS başlığı ile başlar PE başlıkları ile devam eder, bölüm tablosu ve bölümler kısmı ile sona erer. PE dosya formatının başlık olarak adlandırılan kısmında MS DOS stub, PE dosyasının imzası, COFF (CommonObject File Format) başlığı, ve diğer opsiyonel başlıklar mevcuttur. Bölüm tablosu, bölüm adlarını, konumunu ve boyutunu içeren bilgileri barındırır. Bölümler ise gerçek kod parçası, ilk değer atanmış veriler, import tablosu, export tablosu ve kaynakları barındırmaktadır (Alkan vd., 2013). Çalışmada öznitelik olarak kullanılan Opsiyonel Başlık bölümü ise, opsiyonel olarak adlandırılmış olsa da opsiyonel değildir. PE dosya formatının çok önemli bilgilerini barındırmaktadır. Örneğin; AddressOfEntryPoint dosyanın başlangıç noktasının RVA' sını (Relative Virtual Address – Bağıl Sanal Adres), Image Base PE formatının yükleneceği adresi, Section Alignment hafızaya yüklenen bölümlerin konumunu tutar. Import

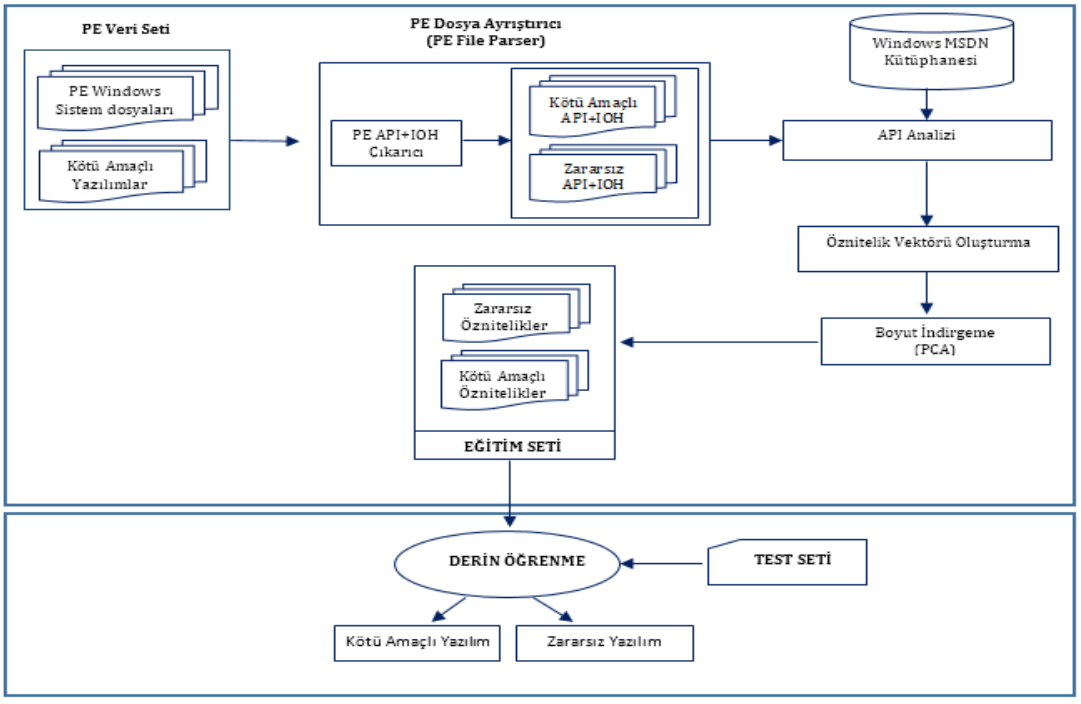
tablosundan elde edilen Windows API çağrıları, yazılımcıların ihtiyaç duyduğu temel fonksiyonları barındıran kütüphaneleri ihtiva etmektedir. Kötü amaçlı yazılımda kullanılan API çağrıları, bu yazılımın davranışları hakkında yorum yapılabilme açısından önemlidir. Örneğin; RegOpenKeyEx API çağrısını kullanan bir yazılım, kayıt defterinde bir değer değiştirmek istiyor ya da bir değer sorgulaması yapıyor yorumunda bulunulabilmektedir.



Şekil 1. PE dosya formatı

Yerel sistemde oluşabilecek zararları önlemek amacıyla oluşturulan sanal bir sistemde, Windows platformunun çalıştırılabilir PE formatındaki dosyaların, IAT (Import Address Table) bölümünden elde edilen Windows API çağrıları ve Opsiyonel Başlık bölümünün özellikleri çalışmada kullanılmıştır.

Elde edilen örnekler öznitelik çıkarma, boyut indirgeme gibi veri madenciliği yöntemleriyle işlendikten sonra bir veri seti oluşturulmuştur. Veri seti DL metoduyla eğitilip teste tabi tutulmuş ve sonuçları değerlendirilmiştir. Kötü amaçlı yazılımları tespit etmek için önerilen tespit mekanizması Şekil 2'de gösterilmiştir.



Şekil 2: Kötü amaçlı yazılım tespit sistemi

2.1. Veri Toplama

Çalışmada Windows işletim sistemini hedefleyen kötü amaçlı yazılım PE dosyaları virusshare.com internet sitesinden indirilmiştir (VirusShare, 2018). Elde edilen örnekler VirusTotal online virüs tarayıcı sitesinde taranmış, kötü amaçlı yazılımların dahil olduğu aile yani hangi türde olduğu tespit edilmiştir (VirusTotal, 2018). Çalışmada kullanılan veri setinde 592 kötü amaçlı yazılım bulunmaktadır. 592 örneğin türleri ve sayıları Çizelge 1'de verilmiştir. Kötü amaçlı olmayan örnekler ise; Windows işletim sisteminin sistem dosyalarından elde edilmiştir. Veri setinde 283 adet zararsız yazılım örneği bulunmaktadır.

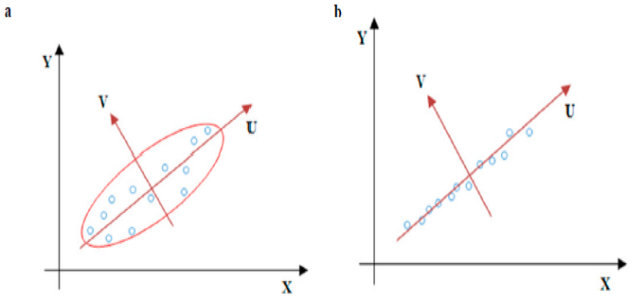
Sıra	Zararlı Türü	Sayı
1	Backdoor	39
2	Email-Worm	67
3	Hacktool	8
4	HEUR:Trojan	103
5	Net-Worm	1
6	P2P-Worm	7
7	Rootkit	10
8	Trojan	151
9	Trojan-Downloader	4
10	Trojan-Dropper	21
11	Trojan.Ransom	7
12	Trojan-Spy	19
13	UDS: DangerousObject.Multi.Generic	83
14	Virus	58
15	Worm	14
	Toplam	**592**

2.2. Öznitelik Çıkarma

PE dosyalarını okumak, API çağrılarını ve Opsiyonel Başlık bölümlerini çıkarmak için yerel sistemden izole edilmiş sanal makine üzerinde, Python programlama dili kullanılmıştır. Python modülü olan Pefile'ı kullanan bir yazılım geliştirilerek API çağrıları ve Opsiyonel Başlık bölümleri çıkarılmıştır. Elde edilen API çağrıları Microsoft'un MSDN kütüphanesinde paylaştığı çağrılarla kontrolü sağlanarak teyit edilmiştir (Microsoft, 2018). Bu sayede farklı import çağrıları elimine edilmiştir. Opsiyonel Başlık bölümleri öznitelik olarak ifade edilirken, bölümün kısa adı ve bu bölümün tuttuğu değer sonuna eklenerek ifade edilmiştir. Örneğin; AddressOfEntryPoint ve tuttuğu RVA 29850 ise AdOfEnPoint29850 şeklinde, MajorOperatingSystemVersion ve dosyanın yükleneceği minimum işletim sistemi versiyonu 1 ise MOSysVer1 kısaltması ile öznitelik olarak eklenmiştir.

2.3. Temel Bileşen Analizi

Temel Bileşen Analizi (PCA), toplam varyansa göre dağıtılmış bir veri kümesinin temel özelliklerini bulmak için istatistiksel bir yaklaşımdır. X-Y koordinat sisteminde bir dizi çok değişkenli dağıtılmış veri verildiğinde, PCA ilk olarak orijinal veri kümelerinin maksimum varyasyonlarını bulur. Bu veri noktaları daha sonra U-V koordinat sistemi adı verilen yeni bir eksen üzerine yansıtılır. U ve V ekseninin yönü ana bileşenler olarak bilinir. Verilerin değiştiği ana yön, ortogonal yönü, V-ekseni ile takip eden U-ekseni ile gösterilir. Şekil 3'teki gibi V eksenindeki tüm veri noktalarının sıfıra çok yakın olması durumunda, veri seti sadece bir U değişkeni ile temsil edilebilir ve değişken V atılabilir (Ng, 2017).

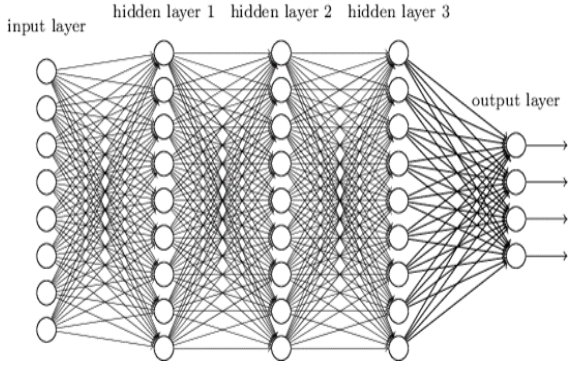


Şekil 3. PCA ile boyut indirgeme (Ng, 2017)

2.4. Derin Öğrenme (Deep Learning)

Derin Öğrenme (Deep Learning-DL), son yıllarda yapay sinir ağları üzerindeki çalışmalardan ortaya çıkan bir makine öğrenme araştırması alanıdır. DL algoritmaları, büyük veri kümeleriyle ölçeklendirilebildikleri ve özellik açısından zengin veri kümelerinden karmaşık kalıpları tanımlamada etkili oldukları için çözümü zor problemlere etkili çözümler sunmaktadırlar. DL, örnek kümeden veri setlerinin temel özelliklerini öğrenme yeteneğine sahiptir (Cui vd., 2018). DL; görüntü işleme (Barros vd., 2017; Ranjan vd, 2017), ses tanıma (Zhang vd., 2017; Zeyer vd., 2017), doğal dil işleme (Young vd., 2017; Mezgec vd., 2018), zaman serisi analizi (Qiu vd., 2017; Siddiqui vd., 2018), kötü amaçlı yazılım tespiti (Hardy vd., 2016; Kolosnjaji vd., 2016; Cui vd., 2018) gibi birçok alanda kullanılmaktadır.

Sinir ağları (NN: Neural Networks) karmaşık problemleri çözmek için doğrusal olmayan ağ yapısını öğrenerek karmaşık fonksiyonları tahmin edebilir. Hatanın geriye doğru yayılarak minimize edildiği Geri yayılım (Backpropagation) algoritmasından daha güçlü olan DL ise insan beyninin öğrenme süreçlerini simüle etmek için Derin Sinir Ağı (DNN: Deep Neural Network) kullanır. DNN, birden çok gizli katmanı olan bir sinir ağıdır. DNN katmanlarında çözülecek problemle ilgili öznitelikler öğrenilir ve bu öğrenilen öz nitelikler takip eden katmanın girdilerini oluşturur. Bu sayede başlangıçtaki katmandan en son katmana doğru özniteliklerin öğrenildiği bir ağ yapısı kurulmuş olur (Ze vd., 2013; Kocadayı vd., 2017; Cui vd., 2018). Bir giriş katmanı, üç tane gizli katman ve bir çıkış katmanına sahip DNN Şekil 4' te verilmiştir.



Şekil 4. Örnek bir DNN (Razzak vd., 2018)

3. Bulgular

Çalışmalar Intel Core i5-4200 CPU 2.5 Ghz işlemci ve 8Gb bellek üzerinde çalışan 64 bit Windows işletim sistemi üzerinde gerçekleştirilmiştir. Uygulamada 875 veriden oluşan 8516 giriş ve bir çıkışa sahip veri seti kullanılmıştır. Burada giriş veri setindeki özellik sayısı çok büyük olduğu için PCA kullanılarak boyut indirgeme yapılmıştır. Veri çeşitli boyutlara indirilerek denemeler yapılmış ve elde edilen sonuçlar Çizelge 2'de verilmiştir.

PCA uygulanırken ve DNN modeli oluşturulurken Python programlama dili ve numpy, pandas, sklearn, keras kütüphaneleri kullanılmıştır.

Öncelikle rastgele olarak verilerin %80'i eğitim %20'si test olarak ayrılmıştır. Model bir giriş katmanı, 2 gizli katman ve 1 çıkış katmanından oluşmaktadır. Birinci gizli katman 10 düğümden, ikinci gizli katman 5 düğümden oluşmaktadır. Giriş katmanı ve gizli katmanlarda aktivasyon fonksiyonu olarak Rectifier Lienar Units (RELU) kullanılmıştır. Çıkış katmanında kullanılan aktivasyon fonksiyonu ise sigmoid'dir. Kurulan ağda başlangıç ağırlıklarının uniform dağılıma göre belirlenmesi amacı ile kernel_initializer parametresi tüm katmanlarda uniform olarak belirlenmiştir. Modeli eğitmeden önce öğrenme sürecinin yapılandırması işlemi compile fonksiyonu yardımı ile gerçekleştirilmiştir. Bu fonksiyona öğrenme hızını ayarlayan fonksiyonu belirlemek için gerekli olan optimizer parametresi olarak adam, model ikili bir sınıflandırma içerdiği için loss parametresi binary_crossentropy ve modelin performansını değerlendirmek için kullanılacak fonksiyonu belirlemek için verilen metrics parametresine accuracy değerleri verilmiştir. Son olarak model fit fonksiyonu kullanılarak eğitilmiştir. fit fonksiyonuna modeli eğitmek için kaç defa çalışacağını belirleyen epoch parametresi 10 ve verileri kaçar kaçar alacağını belirleyen batch_size parametresi 5 olarak verilmiştir.

Elde edilen sonuçlar Çizelge 2'de verilmiştir.

Model No	PCA	Temsil Oranı	Doğruluk
1	150	0.886	1
2	100	0.834	0.994
3	50	0.747	0.988
4	10	0.553	0.977

Çizelge 2. Çalışma sonuçları

Çizelge 2'de de görüldüğü gibi PCA kullanılarak boyut indirgeme işlemi gerçekleştirilmiş ve en iyi sonuç 1 numaralı modelde elde edilmiştir. Bu model'de boyut 150, indirgenen boyutun veriyi açıklama oranı 0.886 ve doğruluk oranı 1'dir.

4. Sonuçlar

Bu çalışmada, Windows PE kötü amaçlı yazılımların tespit edilmesinde Windows API çağrıları ve IOH bölümleri öznitelik olarak kullanılmıştır. Çalışmada kullanılan veri setinin farklı türden kötü amaçlı yazılım ihtiva etmesine dikkat edilmiştir. 15 farklı aileye ait kötü amaçlı yazılım türü veri setinde yer almaktadır. DL metodu ile sezgisel tabanlı bir tespit sistemi oluşturulmuştur. PCA ile 150' ye indirgenen öznitelik sayısı ile % 88,6 temsil etme oranı ve %

100 doğruluk oranına erişilmiştir. Literatürdeki çalışmalar incelendiğinde; günümüzde imza tabanlı ve davranış tabanlı tespit yapan uygulamaların, tespit noktasında yetersiz kaldığı ve eksik noktalarının olduğu görülmüştür. Bu eksikliklerin giderilebilmesi noktasında, başarılı bir şekilde gerçekleştirilen uygulama ve kullanılan yöntem ile yüksek bir doğruluk oranına erişilmiştir.

Kötü amaçlı tespit sistemlerine baktığımız zaman birçok çalışmanın API çağrılarına odaklanmış oldukları görülmektedir. Artık araştırmacılar API çağrılarının yanında farklı özellikler ve farklı yöntemler kullanmaya başlamışlardır. DL ile kötü amaçlı yazılım tespit araştırmalarının da son iki yılda arttığı gözlemlenmiştir. Bu bağlamda gelecek çalışmalarda; farklı özniteliklerin kullanıldığı ya da PE dosyalarının dışında farklı dosya türlerindeki kötü amaçlı yazılımların tespitine yönelik uygulamalar yapılarak, kötü amaçlı yazılımların oluşturabileceği zararların önlenmesi yoluna gidilebilir.

Teşekkürler

Bu çalışma 4537-D1-16 numaralı proje kapsamında Süleyman Demirel Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi tarafından desteklenmiştir.

Kaynaklar

Alkan M., Çifter B., Kılıç ET. (2013). Zararlı Yazılım Tespit, Takip ve Analiz Yöntemleri Geliştirilmesi.6.Uluslararası Bilgi Güvenliği ve Kriptoloji Konferansı, Ankara, Türkiye, 20-21 Eylül 2013.

Barros, P., Parisi, G. I., Weber, C., Wermter, S. (2017). Emotion-Modulated Attention İmproves Expression Recognition: A Deep Learning Model. Neurocomputing, Vol. 253, pp. 104–114.

Basu, I., Sinha, N., Bhagat, D., Goswami, S. (2016). Malware Detection Based on Source Data using Data Mining: A Survey. American Journal Of Advanced Computing, Vol. 3(1). pp. 18-37.

Bazrafshan, Z., Hashemi, H., Fard, S. M. H., Hamzeh, A. (2013). A Survey on Heuristic Malware Detection Techniques. InThe 5th Conference on Information and Knowledge Technology, pp. 113–120, IEEE.

Belaoued, M., Mazouzi, S. (2014). Statistical Study of imported APIs by PE Type Malware. In Advanced Networking Distributed Systems and Applications (INDS) 2014 International Conference on, pp. 82–86, IEEE.

Belaoued, M., Mazouzi, S. (2016). A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features. Journal of Information Processing Systems, Vol. 12(4), pp. 644-660.

Byrd, B., Malik, R., Kandalam, V., Liu, Q. (2014). Malware Detection with Computational Intelligence. In Proceedings on the International Conference on Artificial Intelligence (ICAI), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Las Vegas, USA.

Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G., Chen, J. (2018). Detection of Malicious Code Variants Based on Deep Learning. IEEE Transactions on Industrial Informatics, Vol. 14(7), pp. 3187-3196.

Darshan, S.S., Jaidhar, C.D. (2018). Performance Evaluation of Filter-based Feature Selection Techniques in Classifying Portable Executable Files. Procedia Computer Science, Vol. 125, pp. 346–356.

Gupta, S., Sharma, H., Kaur, S. (2016). Malware Characterization Using Windows API Call Sequences. In International Conference on Security, Privacy, and Applied Cryptography Engineering, 6th International Conference, Hyderabad, India, pp. 271-280, December.

Hardy, W., Chen, L., Hou, S., Ye, Y., Li, X. (2016). DL4MD: A deep learning framework for intelligent malware detection. In Proceedings of the International Conference on Data Mining (DMIN), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Las Vegas, USA, pp. 61-67, July.

Kabakuş, A.T., Doğru, İ.A., Çetin, A. (2015). Android Kötücül Yazılım Tespit ve Koruma Sistemleri. Erciyes Üniversitesi Fen Bilimleri Enstitüsü Fen Bilimleri Dergisi, Vol. 31(1), pp. 9-16.

Kocadayı, Y., Erkaymaz, O., Uzun, R. (2017). Yapay Sinir Ağları ile Tr81 Bölgesi Yıllık Elektrik Enerjisi Tüketiminin Tahmini, Bilge International Journal of Science and Technology Research, 1(1), 59-64.

Kolosnjaji, B., Zarras, A., Webster, G., Eckert, C. (2016). Deep Learning For Classification Of Malware System Call Sequences. In Australasian Joint Conference on Artificial Intelligence, Hobart, Tas, Australia, pp. 137–149, December.

Kumar, A., Kuppusamy, K. S., Aghila, G. (2017). A Learning Model to Detect Maliciousness of Portable Executable Using Integrated Feature Set. Journal of King Saud University-Computer and Information Sciences.

Lim, H. (2016). Detecting Malicious Behaviors of Software through Analysis of API Sequence k-grams. Computer Science and Information Technology, Vol. 4(3), pp. 85-91.

McAfee, https://www.mcafee.com/enterprise/en-us/threat-center/mcafee-labs/reports.html. Erişim Tarihi: 22.5.2018.

Mezgec, S., Eftimov, T., Bucher, T., Seljak, B.K. (2018). Mixed Deep Learning and Natural Language Processing Method for Fake-Food Image Recognition and Standardization To Help Automated Dietary Assessment. Public Health Nutrition, http://doi.org/10.1017/S1368980018000708, pp. 1–10.

Microsoft,https://msdn.microsoft.com/en-us/library/, Erişim Tarihi: 02.03.2018.

Ng, S.C. (2017). Principal Component Analysis to Reduce Dimension on Digital Image. Procedia Computer Science, Vol. 111, pp. 113–119.

Qiu, X., Ren, Y., Suganthan, P.N., Amaratunga, G.A.J, (2017). Empirical Mode Decomposition Based Ensemble Deep Learning for Load Demand Time Series Forecasting, Applied Soft Computing, Vol. 54, pp. 246–255.

Ranjan, R., Patel, V. M., Chellappa, R. (2017). Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Razzak, M.I., Naz, S., Zaib A. (2018). Deep Learning for Medical Image Processing: Overview, Challenges and the Future. Classification in BioApps, Springer, Cham, Vol 26, pp. 323-350

Salehi, Z., Sami, A., Ghiasi, M. (2014). Using feature generation from API calls for malware detection. Computer Fraud Security, 2014(9), pp. 9–18.

Schultz, M. G., Eskin, E., Zadok, F., Stolfo, S.J. (2001). Data Mining Methods for Detection of New Malicious Executables. In Security and Privacy, S&P 2001 Proceedings, 2001 IEEE Symposium on, IEEE, pp. 38–49.

Shabtai, A., Moskovitch, R., Elovici, Y., Glezer, C. (2009). Detection of Malicious Code by Applying Machine Learning Classifiers On Static Features: A State-of-The-Art Survey. Information Security Technical Report, Vol. 14(1), pp. 16–29.

Siddiqui, S.A., Mercier, D., Munir, M., Dengel, A., Ahmed, S. (2018). TSViz: Demystification of Deep Learning Models for Time-Series Analysis. arXiv preprint arXiv:1802.02952.

VirusShare, https://virusshare.com/, Erişim Tarihi: 04.02.2018.

VirusTotal,https://www.virustotal.com/#/home/upload, Erişim Tarihi: 15.03.2018.

W3schools,https://www.w3schools.com/browsers/browsers_os.asp, Erişim Tarihi: 25.05.2018

Wang, C., Pang, J., Zhao, R., Liu, X. (2009). Using API Sequence and Bayes Algorithm to Detect Suspicious Behavior. In: Communication Software and Networks, 2009. ICCSN'09. International Conference on, IEEE, pp. 544–548.

Ye, Y., Li, T., Jiang, Q., Wang, Y. (2010). CIMDS: Adapting Postprocessing Techniques of Associative Classification for Malware Detection. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 40(3), pp. 298–307.

Ye, Y., Wang, D., Li, T., Ye, D., Jiang, Q. (2008). An Intelligent PE-Malware Detection System Based on Association Mining. Journal in Computer Virology, Vol. 4(4), pp. 323–334.

Young, T., Hazarika, D., Poria, S., Cambria, E. (2017). Recent Trends İn Deep Learning Based Natural Language Processing. arXiv preprint arXiv:1708.02709.

Zatloukal, F., Znoj, J. (2017). Malware Detection Based on Multiple PE Headers Identification and Optimization for Specific Types of Files. Journal of Advanced Engineering and Computation, Vol 1(2), pp. 153–161.

Ze, H., Senior, A., Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, pp. 7962–7966.

Zeyer, A., Doetsch, P., Voigtlaender, P., Schlüter, R., Ney, H. (2017). A Comprehensive Study of Deep Bidirectional LSTM Rnns For Acoustic Modeling in Speech Recognition. In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, IEEE, pp. 2462-2466.

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Bengio, C. L. Y., Courville, A. (2017). Towards End-To-End Speech Recognition with Deep Convolutional Neural Networks. arXiv preprint arXiv:1701.02720.