KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

KÖTÜCÜL YAZILIMIN TAŞINABİLİR ÇALIŞTIRILABİLİR DOSYALAR ÜZERİNDE MAKİNE ÖĞRENMESİ YAKLAŞIMLARI İLE TESPİTİ

KANAN MUKHTAROV

KOCAELİ 2023

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

KÖTÜCÜL YAZILIMIN TAŞINABİLİR ÇALIŞTIRILABİLİR DOSYALAR ÜZERİNDE MAKİNE ÖĞRENMESİ YAKLAŞIMLARI İLE TESPİTİ

KANAN MUKHTAROV

Doç.Dr. Halil YİĞİT
Danışman, Kocaeli Üniv. .....
Doç.Dr. Süleyman EKEN
Jüri Üyesi, Kocaeli Üniv. .....
Dr.Öğr.Üyesi Ekin EKİNCİ
Jüri Üyesi, Sakarya Uygulamalı Bilimler Üniv. .....

Tezin Savunulduğu Tarih: 23.01.2023

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez/proje çalışmasında,

- Bu tezin/projenin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin/Projenin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez/proje çalışması olarak sunmadığımı,

beyan ederim.

■ Bu tez/proje çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

☐ Bu tez/proje çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

(İmza)

Kanan MUKHTAROV

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin/projemin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet hakları bende kalacak, tezimin/projemin tamamının ya da bir bölümünün gelecekteki makale, kitap, tebliğ, lisans, patent gibi çalışmalarda kullanımı, danışmanımın isim hakkı saklı kalmak koşuluyla ve her iki tarafın bilgisi dâhilinde bana ait olacaktır.

Tezin/projenin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin/projenin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim kurulu tarafından yayınlanan ***"Lisanüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.

☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.

■ Tezim/projem ile ilgili gizlilik kararı verilmemiştir.

(İmza)

Kanan MUKHTAROV

ÖNSÖZ VE TEŞEKKÜR

Kötü amaçlı yazılımlar günümüzde çok yaygın bir hale gelmiştirler. Kötü amaçlı yazılım, herkes tarafından kullanılan telefon, bilgisayar, internet gibi teknolojileri kullanarak kar elde etmek veya başka bir tarafa zarar vermek için tanımlanmıştır. Her geçen sene saldırıya uğrayan kullanıcı sayısı artmaktadır. Bu nedenle Türkiye de dâhil olmak üzere pek çok ülke bu konuyu ele alıyor.

Kötü amaçlı yazılımları tespit etmek için çeşitli yöntemler vardır. En temel ve basit olanı, dosya veri analizine dayanan statik analizdir. Bu verileri makine öğrenimine uygulayarak iyi sonuçlar elde edilebilir.

Bu çalışmanın gerçekleştirilmesinde, üç sene boyunca değerli bilgilerini benimle paylaşan, her zaman sorun yaşadığımda yardımcı olan, güler yüzünü ve samimiyetini benden esirgemeyen, söylediği her kelimenin hayatıma kattığı önemi asla unutmayacağım kıymetli danışman hocam Doç. Dr. Halil YİĞİT'e sonsuz teşekkürlerimi sunarım. Ayrıca, her zaman destek olan aileme ve arkadaşlarıma teşekkürlerimi sunarım. İyi ki varsınız.

Ocak - 2023 Kanan MUKHTAROV

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

f : Fonksiyon
$Gmin$: Görüntünün minimum piksel değeri
$Gmax$: Görüntünün maksimum piksel değeri
GN : 0 ile 1 arasında normalize edilmiş görüntü

Kısaltmalar

AI : Artificial Intelligence (Yapay Zeka)
ARPANET : Advanced Research Projects Agency Network (Gelişmiş Araştırma Projeleri Dairesi Ağı)
AV : Antivirus (Antivirüs)
CART : Classification and Regression Trees (Sınıflandırma ve Regresyon Ağacı)
CPU : Central Processing Unit (Merkezi İşlem Birimi)
DDOS : DistrubutedDenial of Service Attack (Dağıtılmış Hizmet Reddi)
DLL : Dynamic-link library (Dinamik Bağlantı Kütüphanesi)
DN : Doğru Negatif
DP : Doğru Pozitif
DT : Decission Tree (Karar Ağacı)
GPU : Graphics Processing Unit (Grafik İşlem Birimi)
IP : Internet Protocol (İnternet Protokolü)
ITU : International Telecomunication Union (Uluslararası Telekomünikasyon Birliği)
KNN : K-Nearest Neighbour (K-En yakın Komşu)
LAN : Local Area Network (Yerel Alan Ağı)
LR : Logistic Regression (Lojistik Regresyon)
ML : Machine Learning (Makine Öğrenmesi)
PC : Personal Computer (Kişisel Bilgisayar)
PDA : Personal Digital Assitant (Kişisel Dijital Asistan)
PE : Portable Executable (Taşınabilir Çalıştırılabilir)
RAM : Random Access Memory (Rasgele Erişimli Bellek)
RF : Random Forest (Rastgele Orman)
SCADA : Supervisory Control and Data Acquisition (Merkezi Denetleme Kontrol ve Veri Toplama)
SVM : Support Vector Machine (Destek Vektör Makinesi)
TCP : Transmission Control Protocol (İletim Kontrol Protokolü)
UDP : User Datagram Protocol (Kullanıcı Veribloğu Protokolü)
YN : Yanlış Negatif
YP : Yanlış Pozitif

KÖTÜCÜL YAZILIMIN TAŞINABİLİR ÇALIŞTIRILABİLİR DOSYALAR ÜZERİNDE MAKİNE ÖĞRENMESİ YAKLAŞIMLARI İLE TESPİTİ

ÖZET

Kötü amaçlı yazılımlar, ev kullanıcılarından işletmelere kadar herkes için büyük bir tehlike olmaya devam etmektedir. Günümüzde cihazların birçoğu ağlar üzerinden internete bağlanmaktadır. Bu nedenle, kötü amaçlı yazılımlar kolay ve hızlı bir şekilde yayılabilmektedir. Bu çalışmanın amacı, zararlı Portable Executable (PE) dosyalarını makine öğrenimi (ML) algoritmaları ile tespit etmektir. Sınıflandırmanın verimliliği ve etkinliği, özellik sayısına ve ML algoritmalarına bağlıdır. Bu çalışmada, bilgisayar korsanları tarafından saldırılarda sıklıkla kullanılan bilgisayar virüsü, bilgisayar solucanı, truva atı, reklam yazılımları ve arka kapı yazılımlarına ait PE dosyalarından ve Dinamik Bağlantı Kütüphanelerinden (DLL) çıkarılan 5 özellik incelenmiştir. Veri kümesini oluşturmak için Pe Studio programı kullanılmıştır. Excel dosyası formatına dönüştürülen veriler ML yöntemlerinden Karar Ağaçları, K-En yakın Komşular, Rastgele Orman Sınıflandırıcısı, Destek Vektör Makinesi (SVM) ve Lojistik Regresyon algoritmalarına uygulanarak, yöntemlerin kötücül yazılım sınıflandırma başarımı analiz edilmiştir. Yöntemlerin performansları, doğruluk, kesinlik, duyarlılık ve F1 skor parametreleriyle değerlendirilmiştir. Deneysel sonuçlar, rasgele orman sınıflandırıcısının %92 doğruluk oranı, %93 duyarlılık ve %92 F1 skor değeri ile diğer yöntemlerden öne çıktığını göstermektedir.

Anahtar Kelimeler: Kötücül Yazılım, Makine Öğrenmesi, Sınıflandırma, Yapay Zekâ.

DETECTION OF MALWARE WITH MACHINE LEARNING APPROACHES ON PORTABLE EXECUTABLE FILES

ABSTRACT

Malware remains a major threat to everyone from home users to businesses. Today, most of the devices are connected to the internet via networks. Therefore, malware can spread easily and quickly. The aim of this study is to detect malicious Portable Executable (PE) files with machine learning (ML) algorithms. The efficiency and effectiveness of classification depend on the number of features and ML algorithms exploited. In this study, 5 features extracted from PE files and Dynamic Link Libraries (DLL) of computer viruses, computer worms, Trojan horses, adware and backdoor software, which are frequently used by hackers in attacks, are examined. Pe Studio program is utilized to create the data set. The malware classification performance of the methods is analyzed by applying the data converted to Excel file format to Decision Trees, K-Nearest Neighbors, Random Forest Classifier, Support Vector Machine (SVM) and Logistic Regression algorithms from ML methods. The performances of the methods are evaluated with accuracy, precision, sensitivity and F1 score parameters. Experimental results show that the random forest classifier outperforms other methods with its 92% accuracy, 93% sensitivity and 92% F1 score value.

Keywords: Malware, Machine Learning, Classification, Artificial Intelligence.

1.GİRİŞ

Son yıllarda bilgi teknolojilerinin dramatik bir şekilde artmasıyla bilgisayar, tablet, telefon, kişisel dijital asistan (PDA) gibi cihazların kullanımı da yoğun bir şekilde artmıştır. Kullanıcılar istedikleri yerden ve istedikleri zamanda internet bağlantısı vasıtasıyla bilgiye erişebilmektedirler. Uluslararası Telekomünikasyon Birliği (ITU) 2022'de yaklaşık 5,3 milyar insanın (dünya nüfusunun yüzde 66'sının) İnternet kullandığını tahmin etmektedir (URL-1). Bu, 2019'dan bu yana yüzde 24'lük bir artışa karşılık gelmektedir. Bu süre zarfında yaklaşık 1.1 milyar kişi çevrim içi olmuştur.

İnternet kullanımı oranının artması veri güvenliği zafiyetini de beraberinde getirmektedir. Bu durum saldırgan ve diğer kötü niyetli kişilerin de ilgisini çekmektedir. Saldırganlar geliştirdikleri kötücül yazılımlarla kurum/kuruluş veya kişilerin hassas bilgilerini elde etmeye çalışmaya, bilgisayar alt yapılarını kullanılamaz hale getirmeye, bilgisayara izinsiz erişim hakkı elde etmeyi amaçlamaktadırlar.

Geçmişten günümüze sayısız kötücül yazılımlar kullanılarak saldırılar gerçekleştirilmiştir. 1971 yılında "The Creeper" olarak adlandırılan dünyanın ilk virüs Kavram Kanıtı gerçekleştirildi. Bulaştığı bilgisayarda, kullanıcıya "I'm the creeper, catch me if you can!" mesajını gösterdi. İleri Araştırma Projeleri Ajans Ağı (ARPANET) bilgisayarları üzerinden yayılmıştır. Sistemlere zarar vermeyen ilk kötücül yazılımıdır (Currie ve Fidler, 2015). 'Brain' olarak adlandırılan ilk bilgisayar virüsü Pakistanlı iki kardeş Amjad Farooq Alvi ve Basit Farooq Alvi tarafından geliştirilmiştir (URL-2). Bu virüs disketlerin kopyalanması yoluyla dünya çapında hızla Avrupa ve Kuzey Amerika'ya yayılmıştır. 1999 yılında Happy99 virüsü, Melissa solucanı ve Kak solucanı gibi kötü amaçlı yazılımlar internet üzerinden kullanılan Microsoft ortamları aracılığıyla çok hızlı bir şekilde yayılmıştır (Todd Austin ve diğ,, 2015). 2000 yılında Filipinler'de bir üniversite öğrencisi olan Onel De Guzman tarafından yaratılan VBScript solucanı olan ILOVEYOU saatler içinde milyonlarca Windows makinesine bulaşarak fotoğraflar, ses dosyaları ve belgeler dahil her türlü dosyayı yok etmiştir (Matt Bishop, 2000). 2007'de Estonya, kasıtlı bir DDoS saldırısına maruz kalarak başbakanın sitesinin yanı sıra okullar ve bankalar gibi birkaç devlet kuruluşu

çökmüştür (Christian Czosseck, 2007). 2010'ların başında, denetleyici kontrol ve veri toplama cihazlarına (SCADA) yönelik olarak geliştirilen ilk spesifik ulus-devlet kötü amaçlı yazılımı olan Stuxnet İran'daki kuruluşları, özellikle nükleer tesisleri hedef almıştır (David Kushner, 2013). 2012 yılında öncelikle Orta Doğu'daki kuruluşları hedef alan 'Flame' isimli kötücül yazılım solucan mantığıyla hareket ederek, LAN'lar aracılığıyla yayılmıştır. Bilgisayar ekran görüntülerini ve sesleri kaydetme ve yakalama, klavyede yazılanları loglama, Skype konuşmalarını gizlice dinleme ve kaydetme yeteneğine sahipti. Flame, topladığı bu bilgileri 5 ayrı şifreleme algoritması ile kriptolayarak gizledikten sonra dosyaları önceden belirlenmiş bir sunuculara gönderir (URL-3). 2013'te Bitcoin (BTC) aracılığıyla ödeme talep eden ilk fidye yazılımı CryptoLocker geliştirildi. Şifre çözmenin bedeli olarak değeri 13 ila 1.100 dolar arasında olan iki BTC talep edilmiştir (URL-4). 2016'da, IoT cihazlarını hedefleyen ilk botnet olan Mirai'nin yazılımcıları tarafından, tanınmış Web sitelerine (Twitter, Netflix, Reddit ve GitHub) DDoS saldırısı başlatılmıştır. Dünya çapında bu Web sistelerine erişimi ve hizmetleri kesintiye uğratmıştır (Armağan Keskin ve diğ., 2019). 2017'de Microsoft Windows çalıştıran 230.000'den fazla bilgisayara Eternal blue güvenlik açığından yararlanan WannaCry fidye (ransomware) yazılımı bulaşmıştır (Ehrenfeld, 2017).

Kaspersky Security Network'ün raporuna göre Türkiye'de Truva Atı casuslarından etkilenen kullanıcı sayısı 2022'nin ikinci çeyreğinde ilk çeyreğe göre %5 oranında artmıştır (URL-5). Kaspersky'nin son üç aylık kötü amaçlı yazılım raporunda Microsoft Office açıkları üzerinden saldırıya uğrayan kullanıcı sayısının son çeyrekte %17 artış gösterdiği yer almıştır (URL-6). Kaspersky Security Network'ün kurumsal kullanıcılara yönelik verileri incelendiğinde, arka kapı bilgisayar kötü amaçlı yazılımları tespitlerinin 2022'nin ikinci çeyreğinde %29 azaldığı analiz edilmiştir (URL-7). Ayrıca, Türkiye'deki suistimal tespitleri 2022'nin ikinci çeyreğinde %20 oranında artmıştır (URL-8). Kaspersky analizi, kimlik avı ve dolandırıcılık / sosyal mühendislik yoluyla gerçekleştirilen veri kaybı tehditleriyle ilgili saldırıların 2022'nin ikinci çeyreğinde bir önceki çeyreğe göre %79 oranında önemli ölçüde arttığını göstermektedir (URL-9) Kaspersky Digital Payment anketine göre, Türkiye'de katılımcıların %48'i çevrimiçi bankacılık veya mobil cüzdan hizmetlerini kullanırken kimlik

avı dolandırıcılığıyla karşılaştı. %49'u kişisel olarak sahte web sitelerine denk gelirken, %42'si sosyal mühendislik kullanan dolandırıcılık yöntemlerine maruz kalmıştır (URL-10).

Mevcut antivirüs (AV) yazılımları, bilinen kötü amaçlı yazılımları tespit etmede etkilidir. Ancak, bilinmeyen tehditler AV yazılımlarından kaçabilir. Yapılan bir araştırmada, güncel kötü amaçlı yazılımların yaklaşık yüzde 80'inin en son antivirüs yazılımları tarafından yakalanamadığını göstermiştir (URL-10) . Geleneksel imza tabanlı antivirüs yazılımları belirli bir kötü amaçlı yazılım parçasını algılamak ve imzayı bir kötü amaçlı yazılım veritabanına eklemek için manuel olarak bir imza oluşturur. Antivirüs yazılımı, her yeni tarama sırasında imza veritabanına başvurarak kötücül yazılım için karşılaştırma yapar. İmza tabanlı algılama yöntemi virüsler, truva atları ve solucanlar ilk ortaya çıktığında etkiliydi. Ancak, bilgi teknolojilerinin hızlı bir şekilde gelişmesiyle birlikte kötücül yazılımlar da evrim geçirmektedir. Bu nedenle, AV programlarının imzaları da değişen gelişmiş zararlı yazılımları tespit etmesi zorlaşmaktadır yada yetersiz kalmaktadır. Bu sorunu çözmek için, (Schultz ve diğ., 2001) makine öğrenimi yöntemleri ve sınıflandırıcılarının kullanılması yaklaşımının gerçekten de kötü amaçlı yazılımların başarılı bir şekilde algılanmasına yol açabileceğini göstermiştir.

İki tür kötücül yazılım analiz ve tespit mekanizması vardır. Bunlar statik analiz ve dinamik analiz yöntemleridir. Literatürde yapılan çalışmalar, statik özellik tabanlı analizin daha verimli ve etkili olduğunu göstermektedir (Shijo ve Salim, 2015). Ancak, kötücül dosya (yazılım) paketlendiğinde veya şifrelendiğinde statik analiz gerçekleştirmek zordur. Bu durumda dinamik analiz yönteminin kullanılması gerekmektedir. Dosya paketlenmişse veya şifrelenmişse, önce çalışabilir bir hale getirilmelidir. Dinamik analizin gerekliliklerinden biri, zararlı dosyanın sanal makine (Virtual box, VMware) veya kum havuzu (Sandbox) gibi kontrollü bir ortamda çalıştırılmasıdır.

Bu tez çalışmasında, kötücül yazılımların tespit edilmesi amacıyla statik analiz yöntemleri uygulanmıştır. Az sayıda öz nitelikler kullanılarak zararlı PE (Portable Executable) dosyalarını tespit etmek için çeşitli makine öğrenimi algoritmaları incelenmiştir. Bu algoritmalar birçok alanda faydalanılan Karar Ağaçları, K-En Yakın Komşular, Rastgele

Orman Sınıflandırıcısı, Destek Vektör Makinesi (SVM), Lojistik Regresyon yöntemleridir. Bu yöntemlerin etkinliğini karşılaştırmak için, trojanlar, arka kapılar, reklam yazılımları (adware), solucanlar ve bilgisayar virüsleri kullanılmıştır. Kötücül yazılımları analiz etmek için Pestudio programı ile 3500 veri toplanmıştır. Toplanan verilerin yarısı iyi huylu (benign), diğer yarısı kötücül (malware) dosyalardan oluşmaktadır. Bu çalışmada, kötücül yazılım tespiti için PE dosyalarından çıkarılan virtual_size, raw_size, file_size, entropy, size_of_optional_header öznitelikleri kullanılmıştır. Algoritmaların başarım performansını değerlendirmek için literatürde en yaygın olan doğruluk, kesinlik, duyarlılık ve F1 skoru parametreleri kullanılmıştır. Sonuçlar, en yüksek doğruluk değeri %92 oranı ile Rastgele Orman Sınıflandırıcısı tarafından elde edildiğini göstermiştir. Sonuç olarak, elde edilen veriler aynı öğrenme tekniğinin farklı kötücül yazılım türleri üzerinde benzer performans gösterdiğini ortaya koymuştur. Bu durum, PE dosyalarından çıkarılan öz niteliklerin belirli bir kötücül yazılım türüne karşı önyargılı olmadığını göstermiştir.

Bölüm 2'de, bu konuda yapılan ilgili çalışmalar verilmektedir. Bölüm 3'te, bazı arka plan bilgilerine verilmektedir. Kötü amaçlı yazılım hakkında kısa bilgiler sunulmaktadır. Ek olarak, PE yürütülebilir dosyalarına genel bir bakış olacaktır. Bölüm 4'te, veri analizi tekniklerini ve bunların pratikte nasıl uygulanabileceğini tartışılacaktır. Bölüm 5'te, bu çalışmada kullanılacak makine öğrenimi özelliklerini ve algoritmaları incelenmektedir. Bölüm 6'da araçlar ve yöntemler gözden geçirilecektir. Son olarak Bölüm 7'de, deneyler ve sonuçlar sunulacaktır.

2. İLGİLİ ÇALIŞMALAR

Literatürde, makine öğrenimi algoritmalarını kullanarak PE kötücül yazılım algılama performansını iyileştirmeye yönelik çalışmalar mevcuttur.

Shafiq ve diğerleri (2009) yaptıkları çalışmada, zararlı PE yazılımlarını tespit etmek için PE-Miner programını önerdiler (Shafiq ve diğ., 2009). Toplam 189 parametre kullandılar, bunların 73'ü belirli DLL içe aktarmalarının varlığını veya yokluğunu gösteren ikili parametrelerdi. İyi huylu ve kötücül yürütülebilir dosyaları ayırt etmek için 0,991'lik ROC eğrisi (AUC) altındaki toplam alanı bildirdiler.

Schultz ve diğerleri (2019), yalnızca PE başlık özelliklerini kullananlar da dahil olmak üzere, birkaç tür özellik kullanan deneyler sundu. %89,4 doğrulukla DLL içe aktarmaları ve içe aktarma işlevi içe aktarmalarının bir listesini kullandılar (Schultz ve diğ., 2019). Benzer bir çalışmada, yalnızca içe aktarılan işlevler dikkate alınmış ve %90'dan fazla sınıflandırma doğruluğu elde edilmiştir (Masud ve diğ., 2008).

Elovichi ve diğerleri (2007), içe aktarılan işlevlerin yanı sıra makine tipi ve oluşturma zamanı gibi PE başlık alanlarını kullandılar. Menahem ve arkadaşları (2009), PE başlığındaki özellikleri ve alanları diğer özelliklerle birlikte kullanmıştır (Elovici ve diğ., 2007).

2011 yılında PE-dosyaları için zarar verici yazılımı bulmak için bir yöntem önerilmiştir, graf analizi yöntemine dayanmaktadır (Anderson ve diğ., 2011). Analiz için kullanılan statik özellikler, işlenmemiş ikili dosyalar ve işletim kodlarının kullanımını içeriyordu, n-gram yaklaşımı ile, sırasıyla, emir takip ediyor, kontrol akışı grafikleri ve sistem çağrılarının izlenmesini içeriyordu. Yazarlar eğitim için 776 iyi huylu ve 780 kötücül yazılım örneği kümesi kullanıldı. Sınıflandırma algoritması olarak destek vektör makinesi algoritması kullanılırken, graf kenarları arasındaki benzerlik dizini bulmak için çoklu görev öğrenmesi yaklaşımı kullanıldı.

Hodmoradi ve diğerleri, 2015 yılında metamorfik zararlı programların tespitine ilişkin bir evristik model sunmuştur (Hodmoradi ve diğ., 2015). Yazarlar analiz için statik özellikleri

(opkodları) kullandı. Yazarlar, dosyayı IDA Pro ile dezassemble edip opkodlarının istatistik çıkarımlarını çıkartmak için bir çıkarıcı kullandılar. Çalışmada, iyi niyetli ve kötücül yazılım olarak 1200 örnek sınıflandırılması için kullanıldı. Kötü yazılımın tespiti için: j48, j48graft, LADTree, NBTree, Rastegele Orman Sınıflandırıcısı, REPTree algoritmaları kullanıldı. Yazarlar, tespit sistemlerinin sınıflandırma doğruluğunun kullanılan sınıflandırma yöntemleri ve seçilen dezassemble tarafından etkilendiğini vurguladılar. En iyi sonuca Rastegele Orman Sınıflandırıcı yöntemi kullanılarak alınmıştır (%97).

2016 yılında, Vadrevu ve diğerleri tarafından makine öğrenmesi yöntemine dayanan bir tespit sistemi geliştirildi (Vardevu ve diğ., 2016). Çalışmada, özelliklerin çıkarılması için statik ve dinamik analiz yöntemlerini uygulandı. Altı sınıflandırma algoritması (Rastegele Orman Sınıflandırıcısı, Naive Bayes, J48, Karar ağaçları, IB1 ve NLP) kullanılarak bir sınıflandırıcı modeli geliştirildi ve PE dosyalarından oluşan 3130 veri eğitildi. Teklif edilen sistemin tespit doğruluğu, Rastgele Orman Sınıflandırıcısı algoritması uygulandığında en iyi (% 99,97) performansı gösterdi.

2017 yılında, Balduoni ve diğerleri. Aynı zamanda PE dosyalarından karakter dizilerini, satırları ve başlıkları çıkarmak için statik analiz yöntemlerini kullandılar. Yazarlar, 4783 örnekli bir veri kümesi kullanarak RF sınıflandırma algoritması ile sistemi eğitti. Yapılan çalışma sonucunda % 96 bir doğruluk sonucu alınmıştır.

2019 yılında IoT ortamında zararlı yazılımların tespitinde benzerlik hassaslaştırma algoritmasına dayalı bir metod önerildi (Namanya ve diğ., 2019). Bu metod ile zararlı yazılım örnekleri arasındaki benzerliğin tespitinde PE dosyalarına dayalı bir veri kümesi kullanıldı ve dört farklı hassaslaştırma metodu (PEHash, Imphash, Ssdeep, Ssdeep kaynak bölümü) incelendi. Son olarak, bu hassaslaştırmaların sonuçları bulanık mantık ve belirlilik faktörü modeli gibi kanıt birleştirme yöntemleri kullanılarak birleştirildi.

2021 yılında, bellek adlı bilişim teknikleri kullanılarak kötü amaçlı yazılım tespiti uygulandı (Sihwail ve diğ., 2021). Yazar, kötü amaçlı yazılımı yürütmek için cuckoo sandbox'ı kullanırken, analiz için de volatility aracını kullandı. Bu makalede, deneyde 2502 kötü amaçlı

dosya ve 966 iyi dosya kullanıldı. API çağrısı özelliği, DLL özelliği, işlem işareti özelliği, ayrıcalık özelliği, ağ özelliği kullanıldı.

2019 yılında, denetim altında olan ikili bir sınıflandırıcı kullanılarak PE dosyalrında kötü amaçlı yazılım tespit edildi (Shukla ve diğ., 2019). Veri kümesi %75 kötü amaçlı dosya ve %25 iyi niyetli dosyalar içeriyor. Önerilen modelin sınırlılıkları, yalnızca belirli bir dosyanın kötü amaçlı yazılım olup olmadığını tespit etmesidir; virüsün türü hakkında herhangi bir bilgi sağlamaz ve dosya şifrelenmişse bu model tarafından tespit edilemez.

2019 yılında, Yousaf ve diğerleri veri madenciliği tekniklerini kullanarak statik analiz yaparak Pe dosyaları üzerinden kötü amaçlı yazılım tespit sistemi geliştirdi (Yousaf ve diğ., 2019). Yazar, 60 özellik çıkardı ve bunların 34'ünün kötü amaçlı yazılım tespiti için önemli olduğunu belirledi. Kullanılan yöntem, sistemin kötü amaçlı ve zararsız dosyaları tespit eder ve ayırır.

2021 yılında Kucharska, makine öğrenimi kullanarak kötü amaçlı yazılım tespiti için statik analiz yöntemini kullandılar (Kucharska, 2021). Bu çalışmada özellikler PE dosyalarından alındı ve sistem için girdi olarak bir görüntü oluşturuldu. Yazar, kötü amaçlı kod sınıflandırması için PE yapısı kullanan modülü, görüntü kullanan modülü ve sınıflandırma sistemi performans deneylerini gerçekleştirdi.

2015 yılında, Belaoued ve diğerleri Chi-kare testine dayalı PE dosyalarında kötü amaçlı kod tespiti önerisinde bulundu (Belaoued ve diğ., 2015). Bu çalışmada, yazarlar kötü amaçlı yazılım tespiti için PE dosyası başlık özelliklerini kullandı ve özellikleri belirleme stratejisi olarak chi-kare skoru ve phi katsayılarını kullandı. Toplanan 552 PE dosyasından, 338 kötü amaçlı yazılım ve 214 zararsız olarak ayrılmıştır.

2016 yılında, Mosli ve diğerleri, Virus Share kötü amaçlı yazılım kütüphanesinden 3.130 zararlı programı ve Windows System32 klasörinden 1.157 zararsız program toplanılmıştır (Mosli ve diğ., 2016). Bu çalışmada, Cuckoo, sisteme malware çalıştırma ve analizi için kullanıldı ve dosyaları zararlı ve zararsız olarak etiketlemek için Virus Total kullanıldı. Volatilite, sandbox'tan bellek görüntüsünün çıkarılmasından sonra analiz için kullanılmıştır.

Önce yapılan çalışmalarla ilgili bilgiler tablo 1'de yer almaktadır. Önceki araştırmalara bakıldığında, kötücül yazılımları etkili bir şekilde tespit etmek için doğrudan PE dosyalarından çıkarılan az sayıda statik özellik kullanılarak hiçbir araştırma yapılmamıştır. Bu çalışmanın katkısı, az sayıda özellik ve beş farklı denetimli öğrenme yöntemi kullanarak PE kötücül yazılımlarının tespitini incelemek ve incelemektir.

Tablo 2.1. Daha önce yapılan çalışmaların kısa özeti

Yazar	**Veri kümesi**	**Kullanılan öznitelikler**	**Analiz için kullanılan yöntem**	**Eğitim için kullanılan yöntem**	**Sonuçlar**
Anderson ve diğ.,	Veri kümesi – 15, 561 örnek	CFG'ler, işlem kodları, talimat izleri, sistem aramaları	Statik ve dinamik analiz	Destek Vektör Makineleri (SVM)	Doğruluk oranı - %97
Liang ve diğ.,	200 örnek	Pe dosyaları, API çağrıları	Statik analiz	Jaccard benzerliği	Doğruluk oranı – %70
Raff ve diğ.,	10,868 dosya (Kaggle + Derbin)	Bayt sekansları	Statik analiz	SVM, KNN, Karar Ağaçları, Yapay Zeka	Ortalama doğruluk oranı –%95
Vadrevu ve diğ.,	Kötücül – 1,251,865 Normal - 400,041	Ağ izleri(traces), Pe dosyaları	Statik analiz	Rastegele Orman Sınıflandırıcısı, Naive Bayes, Karar Ağaçları	En iyi doğruluk %97
Balduoni ve diğ.,	4783 örnek	Pe dosyaları başlıklarından alınmıştır	Statik analiz	Rastegele Orman Sınıflandırıcısı	Doğruluk oranı - %96

3. GENEL BİLGİLER

Bölüm 2.1'de veri kümesinde bulunan kötü amaçlı yazılım türleri ile ilgili bilgi verilmiştir. PE yürütülebilir dosyalar 2.2 bölümünde incelenmiştir.

3.1. Genel Kötücül Yazılımlar

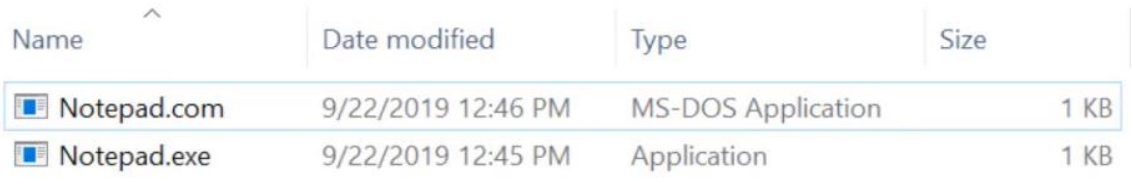
Kötü amaçlı yazılım veya malware (İngilizce: malicious software), bilgisayar ve mobil cihazların çalışmasını bozmak, önemli bilgileri toplamak ve istenmeyen reklamları kullanıcıya göstermek amacı ile kullanılan yazılımdır.

3.1.1. Virüs

Kendiliğinden kopyalar üreten ve onları diğer programlara veya dosyalara ekleyen, görünüşte zararsız görünen başka bir programda genellikle gizlenmiş bir bilgisayar programlarıdır. Genellikle mevcut verileri yok etmek gibi kötü amaçlı eylemlerde bulunur. Bulaştırma yöntemleri:

Virüs-yoldaş (Companion)

1. Yöntem: Virüsun programla aynı ada sahip, ancak COM uzantısına sahip bir dosya oluşturmak.

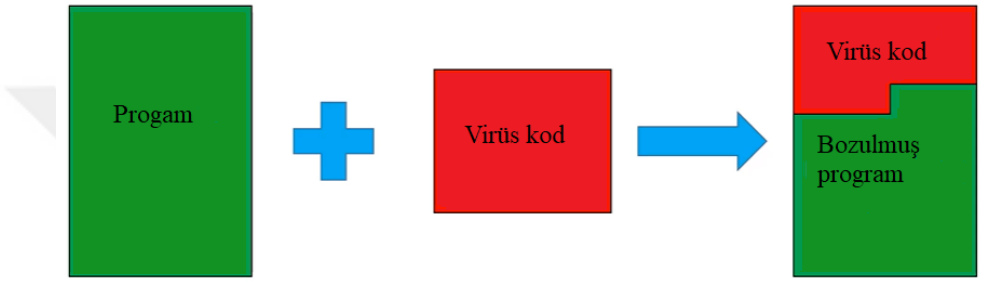


Şekil 3.1. Virüs-yoldaş örneği (URL-11)

Kullanıcı örneğin notepad programını run üzerinden açmak istediği zaman, sistem (Windows) ilk olarak notepad.com programını çalıştırır ve virüs bu şekilde aktif olur. Bu yöntem eski olduğu için şimdi kullanılmamaktadır.

2.Yöntem: İlk yöntemde olduğu gibi programla aynı ada sahip bir dosya oluşturur. EXE uzantılı olarak oluşturulan bu dosya kullanıcı yine run üzerinden çalıştırmak istediği zaman ilk olarak virüsü çalıştırır.

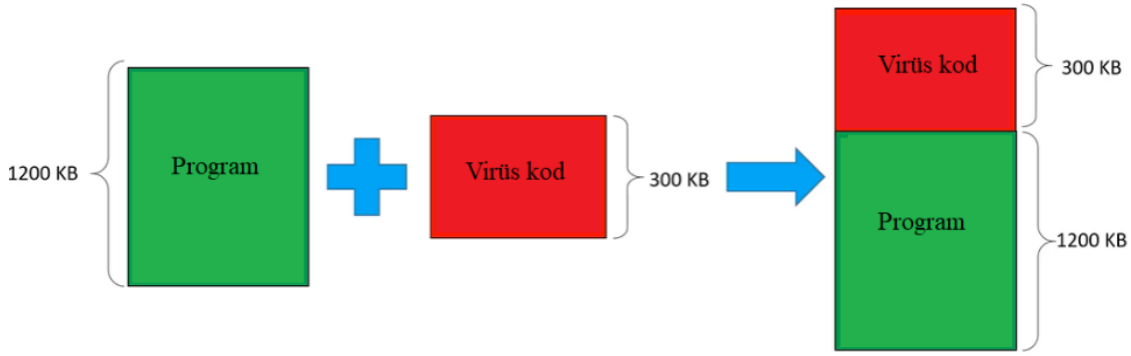
Üzerine Yazma (Overwriting). Şekil 3.2'de gösterildiği gibi virüs tam olarak programın üzerine yazılır. Program çalışmaz hale gelir.



Şekil 3.2. Virüs - üzerine yazma (Overwritting) örneği (URL-11)

Hazırlama (Prepending)

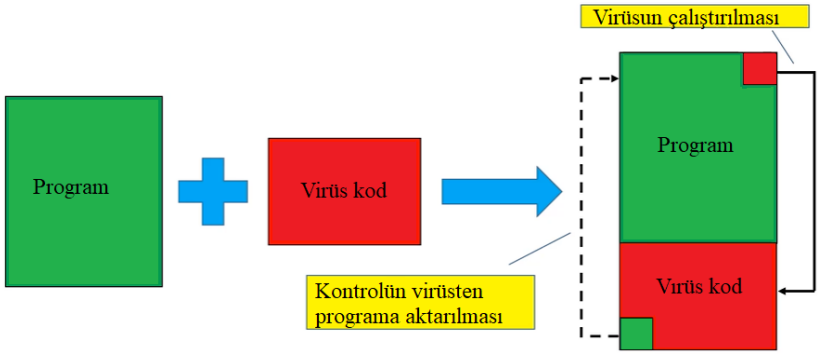
Şekil 3.3'te gösterildiği gibi, virüs programın koduna zarar vermeden üzerine yazılır. Kullanıcı programı çalıştırdığı zaman önce virüs daha sonra program çalışır.



Şekil 3.3. Virüs – Hazırlama (Prepending) örneği (URL-11)

Ekleme (Appending)

Şekil 3.4'te gösterildiği gibi, virüs programın sonuna yazılır. Çalıştırılması için programın başında giriş noktası yaratır. Kendisini çalıştırdıktan sonra programı çalıştırır.



Şekil 3.4. Virüs - Ekleme (Appending) örneği (URL-11)

3.1.2. Truva Atları (Trojan)

Truva atı, genellikle yasal bir yazılım zannedilerek indirilen bir kötü amaçlı yazılım türüdür. Bu isim, Yunanlılar tarafından MÖ 13. yüzyılda Truva'yı ele geçirmek için kullanılan Truva atının yarı efsanevi hikâyesine atıfta bulunmaktadır.

Truva atının klasik bir virüsten temel farkı yayılma yöntemindedir: genellikle sisteme normal, yasal bir program olarak sızar, bu yüzden geleneksel olarak Truva atı olarak adlandırılır. Sızdıktan sonra birçok şey yapabilir: cihaz ve sahibi hakkında bilgi toplamak, bilgisayarda depolanan verileri çalmak, kullanıcı bilgilerine erişimi engellemek, işletim sistemini devre dışı bırakmak vb. (Bowles ve Hernandez-Castro, 2015).

Truva atları, bilgisayar virüslerinden ve solucanlardan farklı olarak kendilerini başka dosyalara enjekte etmeye çalışmazlar. Truva atlarının sınıflandırılması:

- Fidye yazılımı (Ransomware)
- Kriptograflar (Encryptors)
- Yükleyiciler (Downloaders)
- Güvenlik katilleri (Deactivators of protection systems)
- Bank bilgileri çalan (Bankers)
- DDoS Truva Atları (DDOS Trojans)

Fidye yazılım, sisteme veya verilere erişimi engelleyen, kullanıcıyı bilgisayardan dosya silmekle veya kurbanın kişisel verilerini internette yaymakla tehdit eden ve bu tür olumsuz

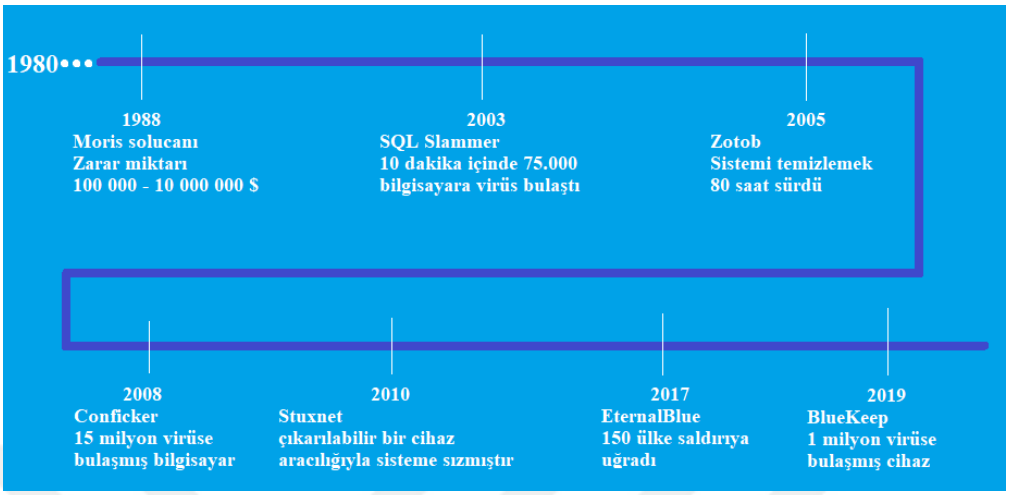
sonuçlardan kaçınmak için fidye ödenmesini gerektiren kötü amaçlı nesne türüdür. Bu davranışın bir örneği WinLock ailesidir.

Kriptograflar, erişimi engelleme aracı olarak kriptografi kullanan gelişmiş bir fidye yazılımı biçimidir. Her zamanki "winlocker" durumunda, kötü amaçlı programı basitçe kaldırmak ve böylece bilgilere yeniden erişim sağlamak mümkün olsaydı, o zaman şifreleyicinin kendisinin yok edilmesi burada hiçbir şey yapmaz, şifrelenmiş dosyalara erişilemez. Ancak bazı durumlarda virüsten koruma yazılımı verileri kurtarabilir. Bir fidye yazılımı örneği CryZip'tir.

İndiriciler, İnternet'ten başka programları veya dosyaları indirmek için tasarlanmış bir Truva atı türüdür. Bir örnek Nemucode'dur. Güvenlik katilleri, anti virüsleri, güvenlik duvarlarını ve diğer güvenlik araçlarını kaldıran veya durduran Truva atlarıdır. Banka bilgileri çalan, banka verilerini (hesap numarası, PIN kodu, CVV, vb.) çalma konusunda uzmanlaşmış bir tür "Truva atı"dır. DDoS Truva Atları (botlar), bilgisayar korsanları tarafından hizmet reddi saldırıları gerçekleştirmek üzere bir botnet oluşturmak için kullanılan kötü amaçlı programlardır. Bir örnek Mirai botnet'tir.

3.1.3. Solucan

Ağ solucanları, kendi kopyalarını oluşturarak yerel ağlar ve İnternet üzerinden yayılabilen bir kötü amaçlı yazılım türüdür. Dosya virüslerinin aksine, ağ solucanları kendilerini çoğaltmak için ağ protokollerini ve cihazlarını kullanabilir. Bu tür kötü amaçlı nesnelerin amacı bir bilgisayara girmek, kendini etkinleştirmek ve kendi kopyalarını diğer kullanıcıların makinelerine göndermektir. Ağ solucanları basit ya da paket solucanlar olabilir. Geleneksel olanlar, bir flash sürücü veya İnternet aracılığıyla sızdıklarında, kendilerini çok sayıda çoğaltır ve daha sonra bu kopyaları bilgisayarda bulunan e-posta adreslerine dağıtır veya yerel ağdaki ortak klasörlere dağıtır. Paket (veya dosyasız) solucanlar belirli bir ağ paketi şeklinde bulunur; bir cihaza sızdıktan sonra, kişisel verileri ve diğer değerli bilgileri toplamak için ana hafizaya (RAM) sızmaya çalışırlar (Sikorski ve Honig, 2012). Bilgisayar solucanların tarihi Şekil 3.5'te gösterilmektedir.



Şekil 3.5. Bilgisayar solucanların tarihi (URL-12)

Ağ solucanları herhangi bir kullanıcının PC'lerini, dizüstü bilgisayarlarını, tabletlerini hedef alır. Bu tür kötü niyetli bir ajanın temel amacı kendi kopyalarını oluşturmak ve daha sonra bunları ağdaki diğer cihazlara yaymak olduğundan, solucanın faaliyetinin sonuçları aşağıdaki gibi olabilir:

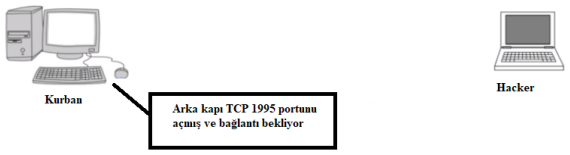
- Bilgisayarın yavaş çalışması
- Azaltılmış sabit disk alanı ve kullanılabilir RAM
- Yabancı dosyaların görünmesi
- Bir programın veya uygulamanın çalışmasıyla ilgili sorunlar
- Hatalar
- Ani kapanma, kendiliğinden yeniden başlatma
- Veri kaybı

3.1.4. Arka Kapı

Arka Kapı, normal kimlik doğrulama prosedürlerini, genellikle İnternet gibi bir ağa yapılan bir bağlantı üzerinden atlamak (bypass) için kullanılan bir yöntemdir. Bir sistem ele geçirildiğinde, gelecekte kullanıcıya görünmez şekilde erişime izin vermek için bir veya daha fazla arka kapı kurulabilir. Bilgisayara solucan, truva atları veya virüsler tarafından

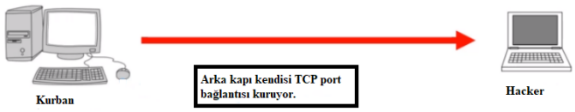
bulaştırılabilir. Sistemde çalıştıktan sonra hackere bağlanmak için ‘Düz’ ve ‘Ters’ olmak üzere iki yöntem kullanabilir.

Düz: Şekil 3.6’da gösterilen durumda kurbanın bilgisayarı hacker tarafından bağlantı bekler. Bu durum saldırı yapan için etkili bir yöntem değil. Güvenlik duvarı tarafından engellenir.



Şekil 3.6. Arka kapı - düz yöntemi (URL-11)

Ters: Bu durumda Şekil 3.7’de görüldüğü gibi kurbanın bilgisayarı hackerle bağlantı kurma için sorgu gönderiyor.

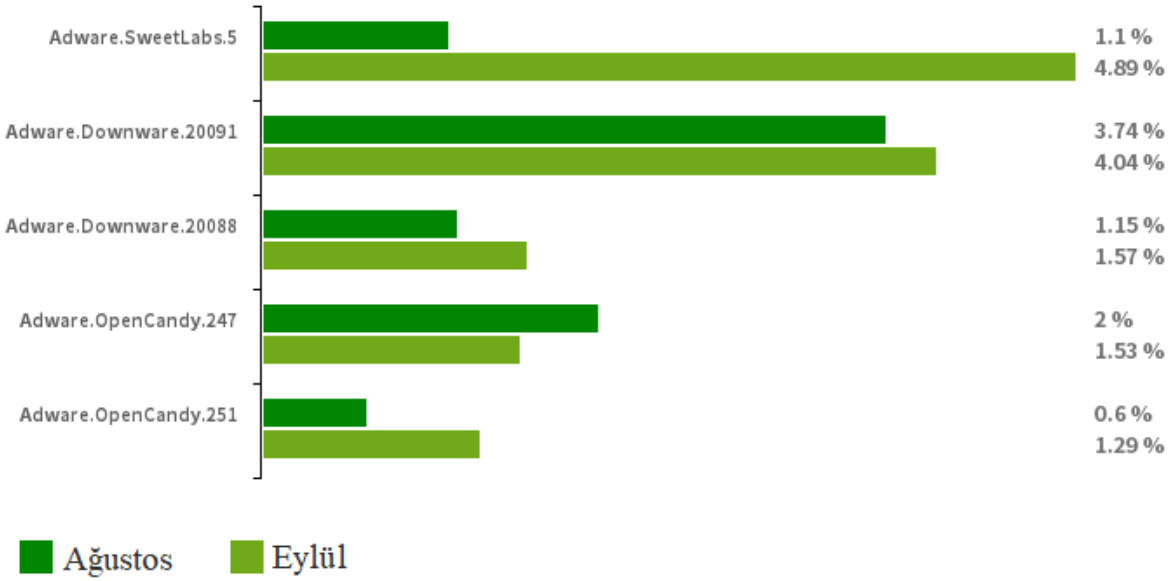


Şekil 3.7. Arka kapı - ters yöntemi (URL-11)

3.1.5. Reklam yazılımları (Adware)

Adware, kullanıcıya reklamlar gösteren ve onun hakkında pazarlama bilgileri toplayan istenmeyen bir programdır. Genellikle ücretsiz indirilen yazılımın kurulum penceresinde görüntülenebilir. Reklam yazılımı, kullanıcının arama istatistiklerini analiz ederek onlar için ‘ilginç’ olan reklamları gösterir. Bir reklam yazılımı ve bir Truva atı arasındaki fark, göreceli olarak zararsız olmasıdır. Reklam yazılımları, kullanıcının ağdaki davranışı hakkında bilgi toplar, ancak bilgisayara zarar vermez ve gizli bilgileri (örneğin, banka kartı bilgileri) çalmaz. Asıl zarar rahatsız edici reklamlardadır: pop-up reklamlar, pankartlar, açılan pencereler. Reklam yazılımından para kazanma, müşteri gösterimler veya tıklamalar için ödeme yaptığında, normal banner reklamlardan para kazanmaya benzer. Genellikle, kullanıcının resmi izniyle bir reklam yazılımı programı yüklenir - örneğin, ek yazılım biçiminde ücretsiz bir uygulama içeren bir pakete. Aynı zamanda, kullanıcı ilgili “tik” e dikkat etmez ve kaldırılabileceğini bilmez veya yazılım yaratıcıları, ürünlerini ücretsiz olarak kullanmayı

teklif eder, ancak reklama tabidir. Reklam yazılımları, virüslü bir sitede sessiz bir kurulum başlatan bir reklama tıklanarak alınabilir. Bu gibi durumlarda, kullanıcının izni olmadan yüklendiği ve kaldırıcısı olmadığı için reklam yazılımı casus yazılım olarak kabul edilir (O'Hanley, 2015). Ağustos – Eylül (2022) ayı için olan Reklam yazılımları kötücül yazılımın yaygınlık oranı Şekil 3.8'de gösterilmiştir.



Şekil 3.8. Reklam yazılımların yaygınlık oranı (URL-13)

3.1.6. Kötücül yazılımların toplanması

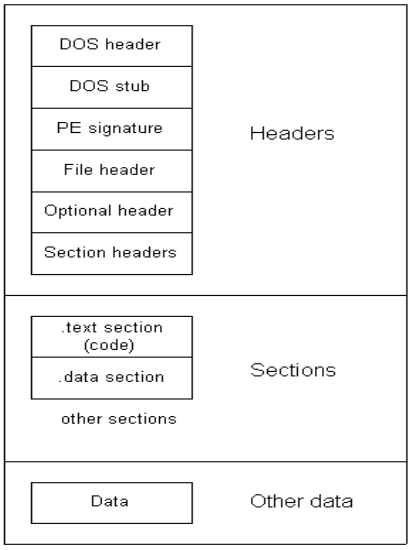
Tüm kötü amaçlı yazılımlar internetten indirildi. Bu dosyaları indirmek için MalwareBazaar, VirusSign, Malshare gibi siteler kullanılmıştır. Bu siteler ayrıca her dosyanın ayrıntılı bir açıklamasını da sunmaktadır. Şekil 3.9'da bir örnek verilmektedir. Dosyalar .dll ve .exe formatlarında indirilir. Daha fazla analiz için, indirilen dosyalar izole edilmiş bir ortama aktarılır (örneğin, VirtualBox'ta oluşturulan bir ortam). Bölüm 4'te, indirilen dosyalar üzerinde yapılan çalışmalar detaylı olarak gösterilmektedir.

SHA1 hash:	e757406bffee53c120a91fc4ce531371a45e4810
MD5 hash:	0392d48e53f663e0f4a847f350314c05
humanhash:	vegan-snake-hawaii-kilo
File name:	0392d48e53f663e0f4a847f350314c05
Download:	download sample
File size:	1'387'008 bytes
First seen:	2023-02-21 02:10:07 UTC
Last seen:	2023-02-21 05:28:39 UTC
File type:	exe
MIME type:	application/x-dosexec
imphash	edc6c3de5dcea9514b11905219670b4c
ssdeep	24576:slxObwiok8Ty80JGTom2s8iQ+FVu1YteAqE9DpQGjl314ZD3bQ2jgcMFS6:slgw5m8PMEb5F4IeAvUGhF4l02oF
Threatray	366 similar samples on MalwareBazaar
TLSH	T1A655124DB29608F9D41BD23A8A869725F2F3741953B443CF57248B769F0A7F2263E342
TrID	44.4% (.EXE) Win64 Executable (generic) (10523/12/4) 21.3% (.EXE) Win16 NE executable (generic) (5038/12/1) 8.7% (.ICL) Windows Icons Library (generic) (2059/9) 8.5% (.EXE) OS/2 Executable (generic) (2029/13) 8.4% (.EXE) Generic Win/DOS Executable (2002/3)
Reporter	@zbetcheckin
Tags:	exe trojan

Şekil 3.9. Tarayıcı üzerinden indirilen kötücül yazılım örneği (URL-11)

3.2. Pe Çalıştırılabilir Dosyalar

PE çalıştırılabilir dosyalar, tüm 32 bit ve 64 bit Windows sistemleri için yürütülebilir bir dosya biçimidir. iki PE dosya biçimi vardır: PE32 ve PE32+. PE32, x86 sistemleri için PE32+, x64 dosya formatıdır. İstenilen bir dosya bir bayt dizisidir.



Şekil 3.10. Pe yürütülebilir dosyasının yapısı (URL-14)

Şekil 3.10'dan görüldüğü gibi Pe yürütülebilir dosya başlıklardan ve bölümlerden oluşmaktadır. PE dosyası MS_DOS başlığı ile başlar, baytlar Şekil 3.10'da gösterildiği gibi MZ'nin ascii değerini alırlar (birinci satır). Bu, dosyanın ilk bölümünü MS_DOS başlık bölümü olarak tanımlar. Sonraki bölüm, tam teşekküllü bir MS_DOS programı olan stub programıdır. Bu programın amacı, program MS_DOS'ta veya Windows NT 3.1'den önceki herhangi bir Windows sürümünde çalıştırıldığında kullanıcıya açıklayıcı bir hata mesajı vermektir. Şekil 3.11'deki hex görünümü "Bu program DOS modunda çalıştırılamaz" mesajını göstermektedir.

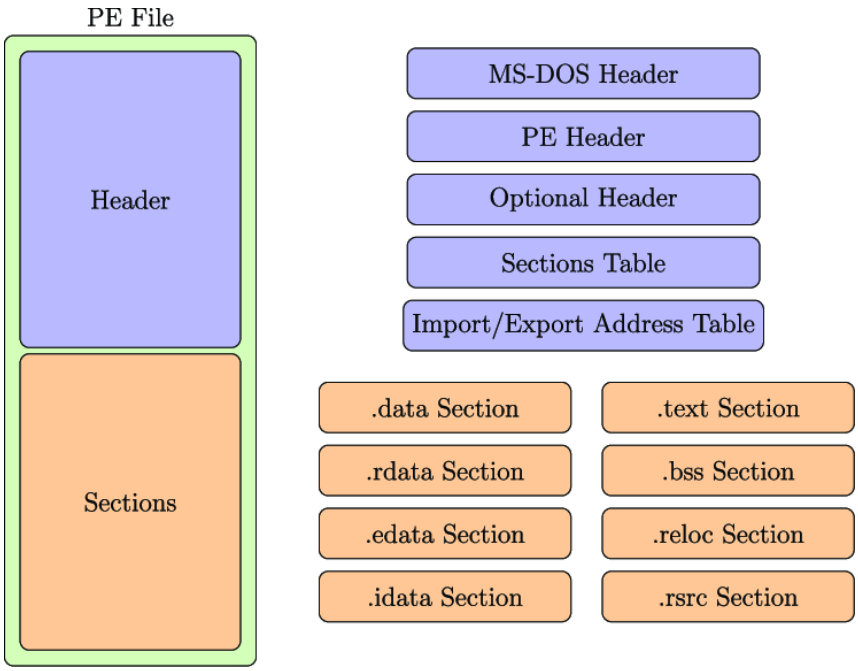
Onaltılık 0x3c'deki adres PE imzasının ofsetini içerir (URL-14). PE imzası MS_DOS saplamasını takip eder ve görüntü biçimini tanımlayan 4 bayttan oluşur; PE formatlı bir görüntü dosyası P, E harflerinden ve iki boş bayttan oluşan bir imzaya sahip olacaktır: yani, "PE\0\0" (URL-14). İmzadan sonra PE başlığı gelir. PE başlığına COFF dosya başlığı da dahildir. Bu, makine alanını içerir, bizim durumumuzda bu INTEL 386 veya uyumlu bir işlemci, Bölüm Sayısı (the NumberOfSections), İsteğe Bağlı Başlık Boyutu (the SizeOfOptionalHeader), Sembol Sayısı (the NumberOfSymbols) ve Sembol Tablosuna İşaretçi (PointerToSymbolTable) olacaktır.

Şekil.3.11. PE dosyasının ilk bölümünün hex görünümü (URL-15)

COFF dosya başlığından sonra İsteğe Bağlı (the OptionalHeader) başlığı gelir. Bu başlık iki bölüme ayrılır: Windows'a özel alanlar ve veri dizinleri. Windows'a özel alanlar işletim sistemi tarafından dosyayı yüklemek ve çalıştırmak için kullanılır. Veri dizinleri, işletim sistemi tarafından gerekli olan farklı tabloların konumunu (RVA) ve boyutunu içerir, bunlar çalışma zamanında belleğe yüklenir.

İsteğe bağlı başlığından (the OptionalHeader) sonra bölüm tabloları gelir. Bunlar temel olarak her veri bölümü için Bölüm başlığıdır. Matt Pietrek bunu "görüntüdeki her bölüm hakkında bilgi içeren telefon rehberi" olarak adlandırır (Pietrek, 1994). Boyut ve konum bilgilerini içerir.

Bölümler (Sections), kod veya verilerin bulunduğu yerlerdir. Örneğin: ".text" bölümü, programın oluşturulduğu eylemleri gerçekleştiren program kodunu içerir, ".data" bölümü, programın çalışması için gereken bazı değişkenleri ve verileri içerir, ".rsrc" bölümü ise kaynak nesneleri ve dosyaları (örneğin: resimler, simgeler, metinler, vb.) içerir. Şekil 3.12'de çalıştırılabilir dosyanın içeriği detaylı olarak gösterilmiştir.



Şekil 3.12. Pe dosya formatı (URL-14)

4. KÖTÜCÜL YAZILIMIN ANALİZ YÖNTEMLERİ

Bu bölümde, kötücül yazılımın analiz yöntemleri sunulmaktadır. Veri analizinde kullanılan veriler, statik analiz yöntemini kullanılarak toplanılmıştır.

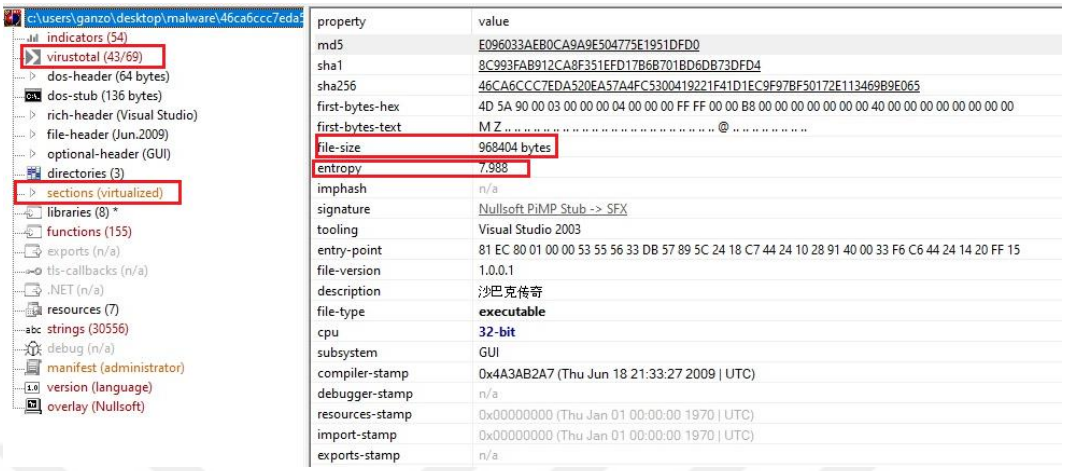
Çoğu zaman, kötü amaçlı yazılımları analiz ederken elimizde yalnızca bir ikili dosya, yani ikili biçimde derlenmiş yürütülebilir bir dosya veya kütüphane olur. Bu dosyanın veya daha doğrusu ikili kodunun nasıl çalıştığını anlamak için özel araçlar ve teknikler kullanılması gerekmektedir. Kötü amaçlı yazılım analizine yönelik iki ana yaklaşım vardır: statik ve dinamik analiz. Tez çalışmasında statik analiz yöntemi kullanılmıştır.

4.1. Statik Analiz

Statik analiz, gerçek talimatlara bakmadan yürütülebilir dosyayı incelemekle ilgilidir (Shijo ve Salim, 2015) Statik analiz, bir dosyanın kötü amaçlı olup olmadığını doğrulayabilir, işlevselliği hakkında bilgi sağlayabilir ve bazen basit ağ imzaları oluşturmanıza olanak tanıyan bilgiler sağlayabilir. Statik analiz basittir ve hızlı olabilir, ancak karmaşık kötü amaçlı yazılımlara karşı pratik olarak etkisizdir ve önemli davranışları gözden kaçırabilir. En temel ve yaygın olarak kullanılan yardımcı programlar: PEiD, Dependency Walker, PEview, Resource Hacker, PeStudio programlarıdır. Tez çalışmasında PeStudio programı kullanılarak veri kümesi için öznitelikler toplanılmıştır.

4.1.1. PeStudio

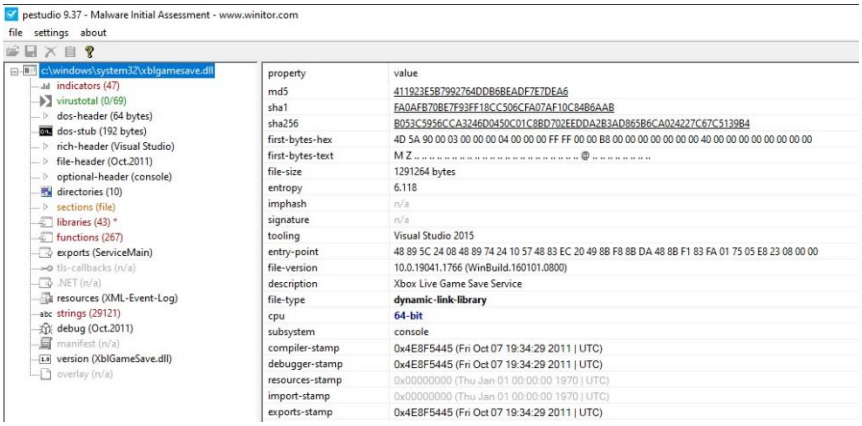
PeStudio, kullanıcılara özellikle Windows işletim sistemlerinde çalışan yürütülebilir dosyaların ayrıntılı bir analizini sağlayan bir araçtır. Yazılım yürütülebilir dosyaları analiz eder ve dosyanın özellikleri, karakteristikleri ve potansiyel riskleri hakkında bilgi sağlar. Bir analist, bu aracı kullanarak, kötü amaçlı yazılım oluşturucuları tarafından kötü amaçlı etkinlikler için yaygın olarak kullanılan işlevleri kolayca tespit edebilir. Bu çalışmada Pestudio programı kullanılmaktadır. Şekil 4.1'de PeStudio uygulamasından bir örnek gösterilmektedir.



Şekil 4.1. PeStudio kullanımı

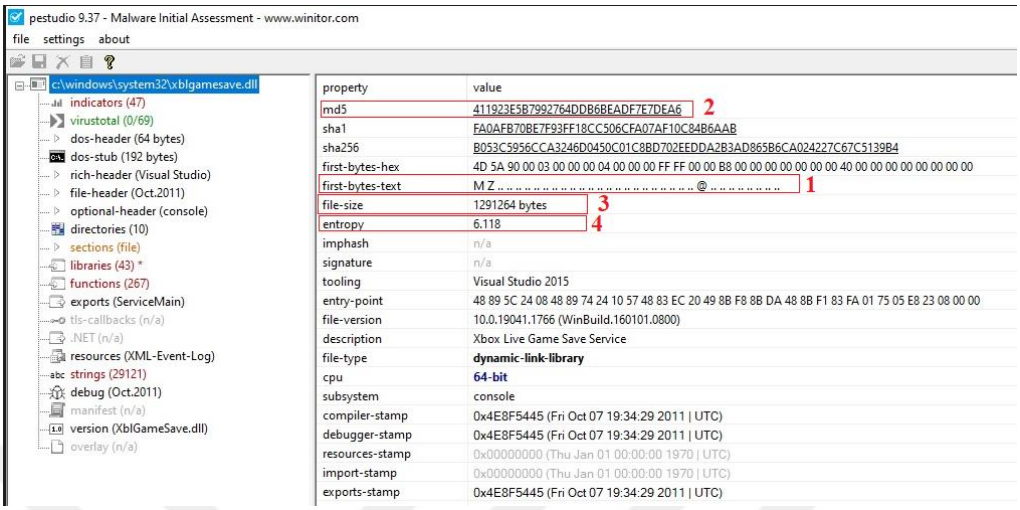
4.2. Statik Analiz Yöntemi İle Veri Toplanması

Veri kümesinde bulunan öznitelikler PeStudio programının yardımı ile toplanmıştır. Veriler bir csv dosyası halinde düzenlenmiştir. Bu program üzerinden alınan öznitelikler aşağıda incelenmektedir. PeStudio programına istenilen .exe veya .dll formatında bir dosya yüklenir, yükleme sonucu Şekil 4.2'de gösterilmiştir.



Şekil 4.2. Bir .exe dosya örneği

Dosyanın çalıştırılabilir olmasını gösteren parametre, first-bytes-text parametresidir (Şekil 4.3'te 1 olarak işaretlenmektedir). Bu parametre MZ'ye eşit değilse, dosya yüklenmeyecektir. Şekil 4.3'te incelenmesi gereken diğer parametreler dosyanın hash değeri (2), dosya boyutu (3) ve entropisidir (4).



Şekil. 4.3. Analiz için önemli parametreler

Hash (md5) - hash, kriptografik algoritmalar tarafından oluşturulan bir karakter dizisidir. Genellikle bu dizeler, giriş verilerinin boyutundan bağımsız olarak sabit bir uzunluğa sahiptir. Örneğin, giriş verisi ‘‘Siyah’’ ve ‘‘Siyah-Beyaz, Beşiktaş’’ aynı uzunlukta hash ismine sahip olacaklar.

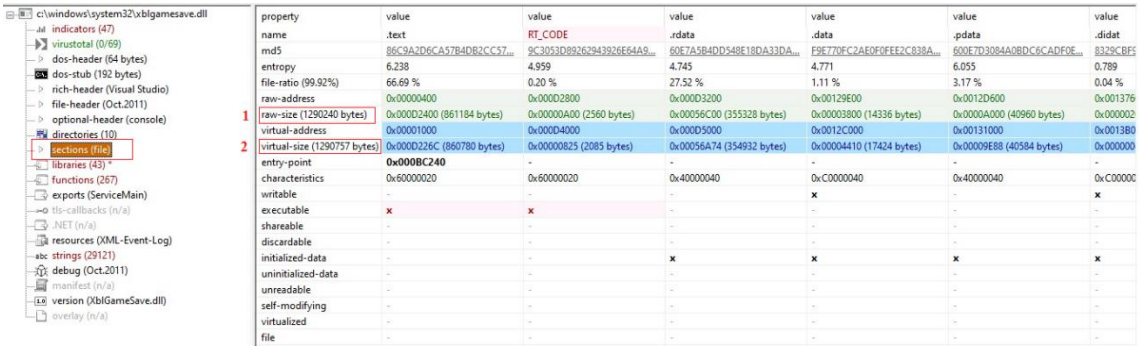
File-size – dosyanın boyutunu gösteren bir parametredir.

Entopy - dosyalardaki bir ikili dizinin rasgele dağılımının ve tahmin edilebilirliğinin bir göstergesidir. Entropi, Shannon formulunü kullanarak, denklem 4.1’de gösterilmiştir, 0 – 8 arasında bir değer alır. Aldığı değer ne kadar büyük olursa kötücül yazılım olma olasılığı bir o kadar artar.

$$H(f) = -\sum_{i=1}^{n} p(i) log_2 p(i) \quad (4.1)$$

Burada *f*- PE dosyasının ikili gösterimi, $p(i)$ ise toplam *n* ikili değere sahip *f* ikili dosyasının *i* biriminin olasılığını gösterir.

Tezde kullanılan diğer öznitelikler Şekil 4.4’te gösterilmektedir.



property	value	value	value	value	value	value
name	.text	RT_CODE	.rdata	.data	.pdata	.didat
md5	86C9A2D6CA57B4DB2CC57...	9C3053D89262943926E64A9...	60E7A5B4DD548E18DA33DA...	F9E770FC2AE0F0FEE2C838A...	600E7D3084A0BDC6CADF0E...	8329CBF5
entropy	6.238	4.959	4.745	4.771	6.055	0.789
file-ratio (99.92%)	66.69 %	0.20 %	27.52 %	1.11 %	3.17 %	0.04 %
raw-address	0x00000400	0x000D2800	0x000D3200	0x00129E00	0x0012D600	0x001376
raw-size (1290240 bytes)	0x000D2400 (861184 bytes)	0x00000A00 (2560 bytes)	0x00056C00 (355328 bytes)	0x00003800 (14336 bytes)	0x0000A000 (40960 bytes)	0x000002
virtual-address	0x00001000	0x000D4000	0x000D5000	0x0012C000	0x00131000	0x0013B0
virtual-size (1290757 bytes)	0x000D226C (860780 bytes)	0x00000825 (2085 bytes)	0x00056A74 (354932 bytes)	0x00004410 (17424 bytes)	0x00009E88 (40584 bytes)	0x000000
entry-point	**0x000BC240**	-	-	-	-	-
characteristics	0x60000020	0x60000020	0x40000040	0xC0000040	0x40000040	0xC00000
writable	-	-	-	x	-	x
executable	x	x	-	-	-	-
shareable	-	-	-	-	-	-
discardable	-	-	-	-	-	-
initialized-data	-	-	x	x	x	x
uninitialized-data	-	-	-	-	-	-
unreadable	-	-	-	-	-	-
self-modifying	-	-	-	-	-	-
virtualized	-	-	-	-	-	-
file	-	-	-	-	-	-

Şekil 4.4. Sections bölümünden alınan değerler

Sections bölümünde incelenesi değerler sanal boyut (Virtual-size) ve ham boyuttur (Raw-size). Ham boyut (Raw_Size) – Diğer adı Size_of_Raw_Data' dır. Ham boyut (Raw_Size) bölümün disk üzerindeki boyut anlamına gelir. Virtual_Size - Sanal bellek boyutu. Nesne yüklendiğinde ayrılacak olan nesnenin boyutudur.

5. KÖTÜCÜL YAZILIMIN TESPİTİ İÇİN KULLANILAN MAKİNE ÖĞRENMESİ ALGORİTMALARI

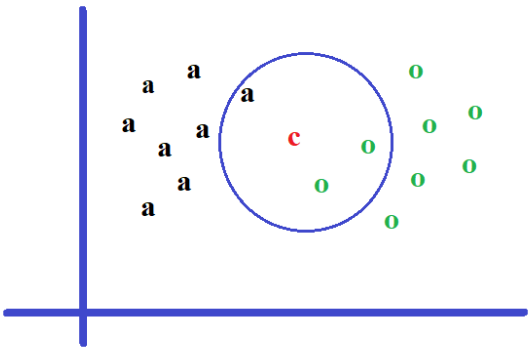
Bu bölümde, tezde kullanılan makine öğrenmesi algoritmalarına kısa bir genel bakış sunulmaktadır.

5.1. Makine Öğrenmesi

Makine öğreniminin amacı, girilen verilerle sonucu tahmin etmektir. Giriş verileri ne kadar çeşitli olursa, makinenin doğru sonuç bulması o kadar kolay olur. Bir makineyi eğitmek için, üç parametreye ihtiyaç vardır. Bunlar; veriler, özellikler (Features) ve algoritma.

5.1.1. K - en Yakın Komşular (K-Nearest Neighbors, KNN)

K-NN, bir sınıflandırma belirlemek için bir eğitim seti kullanan bir kümeleme makine öğrenimi algoritmasıdır (URL-16). K-NN'deki K, mesafe formülünü kullanarak özelliğe göre *K* en yakın komşuyu seçtiğimiz anlamına gelir.



Şekil 5.1. KNN algoritması örneği (URL-16)

Şekil 5.1.'deki örnek, iki özelliğe dayalı ‘‘a’’ ve ‘‘o’’ olmak üzere iki sınıfın grafiğini göstermektedir. ‘‘a’’ ve ‘‘o’’ noktaları eğitim kümesindendir, dolayısıyla ‘‘a’’ ve ‘‘o’’ sınıfından 8 veri olmak üzere toplam 16 element vardır. ‘‘c’’ vektörü, sınıflandırılan girdidir. Örnek : $K = 3$ olarak alındığı zaman, ‘‘c’’ değerinin en yakın 3 komşusu 2 ‘‘o’’ ve 1 ‘‘a’’ olacaktır, çoğunluk kuralına göre bu ‘‘c’’yi ‘‘o’’ sınıfı yapar ‘.

K değeri çift sayıda alındığında, o zaman bazen yeni girilen veriyi hangi sınıfa eklenir diye sorun yaşanır (Örneğin K =4, iki yakın komşu ‘‘o’’ diğer iki komşu ‘‘a’’ olduğunda). Bu sorunun çözmenin bir kaç çeşitli yolları vardır:

- Her zaman *K* değerlerini tek adet olarak seçilebilir
- Eşitlik durumunda, *K* değerini 1- birim azaltılabilir
- Rastgele iki sınıftan biri seçilebilir
- En yakın nokta seçilebilir

Örnek olarak bu tür durumlarda, kullanılan Scikit-Learn kütüphanesi, en yakın noktayı seçilmektedir.

KNN algoritmasının avantajları:

- Algoritma basit ve uygulanması kolaydır
- Bir model oluşturmaya, çeşitli parametreleri ayarlamaya veya ek varsayımlar oluşturmaya gerek yoktur
- Algoritma üniversaldır. Her iki tür sorular için de kullanılabilir: sınıflandırma ve regresyon

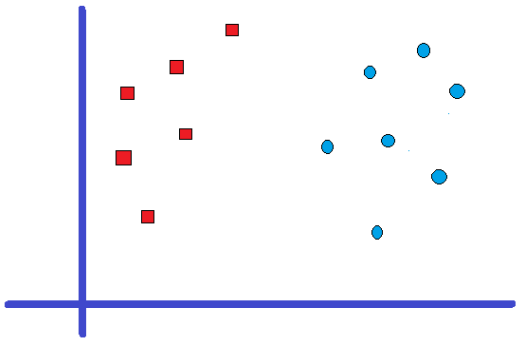
KNN algoritmasının dezavantajları:

- Her zaman k değeri için, en uygun değeri belirlemek gerekir
- Bağımsız değişkenler veya örnek sayısı arttığında algoritmanın çalışması yavaşlar

5.1.2. Destek Vektör Makinesi (Support Vector Machine, SVM)

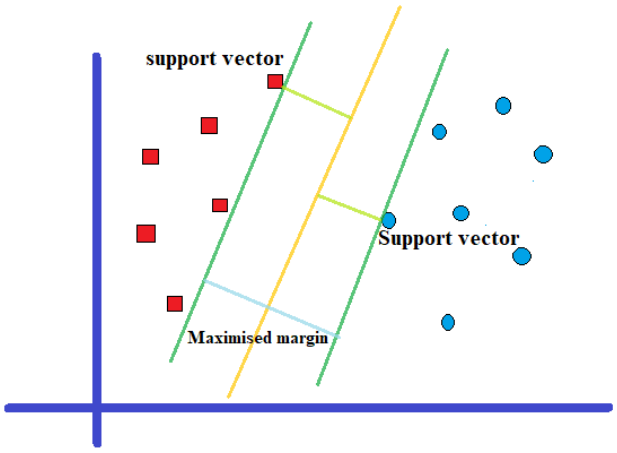
Destek Vektör Makinesi, sınıflandırma ve regresyon problemlerinde kullanılan doğrusal bir algoritmadır. Bu algoritma pratikte geniş bir uygulamaya sahiptir ve hem doğrusal hem de doğrusal olmayan problemleri çözebilir. Destek Vektör Makinesinin çalışması basittir: algoritma, verileri sınıflara bölen bir çizgi veya hiper düzlem oluşturur. Algoritmanın ana görevi, verileri iki sınıfa ayırarak en doğru çizgiyi veya hiperdüzlemi bulmaktır (URL-17).

Şekil 5.2'de verilen örnek üzerinden bakılırsa; bir veri kümesinde kırmızı kareleri mavi dairelerden sınıflandırmak ve ayırmak gerekiyor (kötücül ve normal olarak düşünülebilir). Bu görevdeki asıl amaç, bu iki sınıfı ayıran "ideal" çizgiyi bulmak olacaktır.

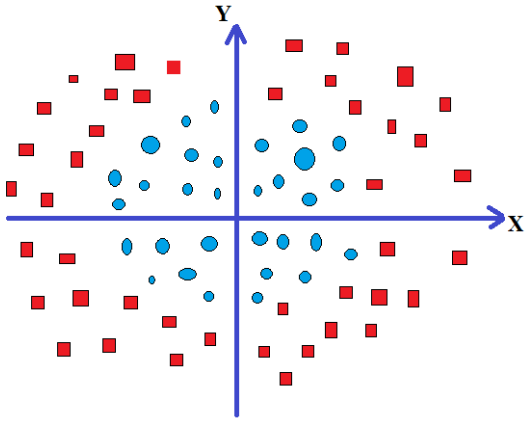


Şekil 5.2. Basit veri kümesi örneği (URL-17)

SVM algoritması Şekil 5.3'te gösterildiği gibi grafik üzerindeki ayrım çizgisine en yakın noktaları arayacak şekilde tasarlanmıştır. Bu en yakın noktalar destek vektörleri (support vector) olarak adlandırılır. Ardından, algoritma destek vektörleri ile ayırma düzlemi arasındaki mesafeyi hesaplar. Bu mesafeye boşluk (margin) denir. Algoritmanın temel amacı boşluk mesafesini maksimize (maximised margin) etmektir. En iyi hiperdüzlem, bu boşluğun mümkün olduğu kadar büyük olduğu hiperdüzlemdir.



Şekil 5.3. SVM algoritması örneği (URL-17)

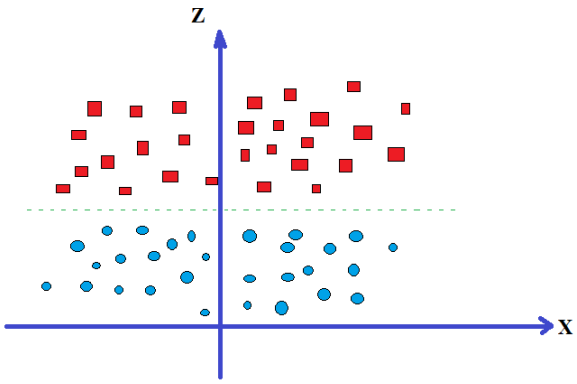


Şekil 5.4. Çekirdek hilesi (Kernel trick) örneği (URL-17)

Şekil 5.4'te verilen örnek incelenecek olursa, bu veri kümesi doğrusal olarak bölünemez. Bu verileri sınıflandırmak için düz bir çizgi çizilemez. Ancak bu veri kümesini, Z ekseni bölebilir.

$$z = x^2 + y^2 \tag{5.1}$$

Şekil 5.5'te aynı veri kümesinin Z ekseninde görselleştirmesi yer almaktadır.



Şekil 5.5. Çekirdek hilesine (Kernel trick) dönüşümü (URL-17)

Yukarıdaki grafikten tüm z değerleri her zaman pozitif olacak, çünkü x ve y'nin karelerinin toplamıdır.

SVM algoritmasının avantajları:

- Küçük boyutlu verilerle iyi çalışır
- Algoritması bölme aralığını maksimize eder sınıflandırma hatalarının sayısını azaltır

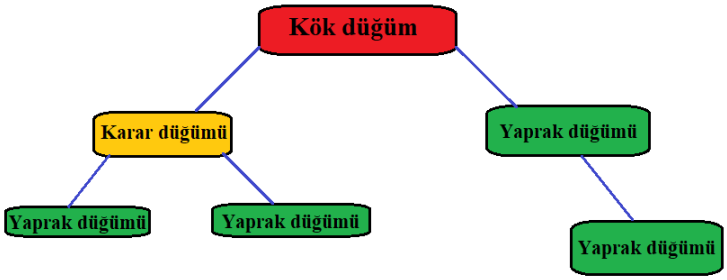
SVM algoritmasının dezavantajları:

- Uzun öğrenme sürelesi (büyük veri setleri için)
- Gürültüye karşı dengesiz: eğitim verilerindeki aykırı değerler referans bozucu nesneler haline gelir ve bölme hiper düzleminin yapısına doğrudan etkiler

5.1.3. Karar Ağaçları (Desicion Tree, DT)

Karar ağaçları, sınıflandırma ve regresyon problemlerini çözmek için veri analizinde kullanılan etkili araçlardan biridir (James ve diğ., 2013). “Eğer ..., o zaman ...” şeklinde belirleyici kurallardan oluşan hiyerarşik ağaç yapılarıdır. Kurallar, eğitim seti üzerindeki öğrenme süreci sırasında otomatik olarak oluşturulur (örneğin, "Eğer satış hacmi 1000 adetten fazlaysa, o zaman ürün iyi satılıyor demektir").

Eğitim setinde, karar ağacı için bir hedef değer belirtilmelidir, çünkü karar ağaçları denetimli öğrenme (supervised learning) temelinde oluşturulan modellerdir. Bu durumda, hedef değişken ayrıksa (sınıf işareti) model sınıflandırma ağacı, sürekli ise regresyon ağacı olarak adlandırılır. Karar ağacı, Şekil 5.6’da gösterildiği gibi, karar düğümleri (desicion node) ve yapraklar (leaf nodes) olmak üzere iki tür öğeden oluşan hiyerarşik yapıda oluşan bir yöntemdir**.** İlk düğüme kök düğüm (root node) denir.



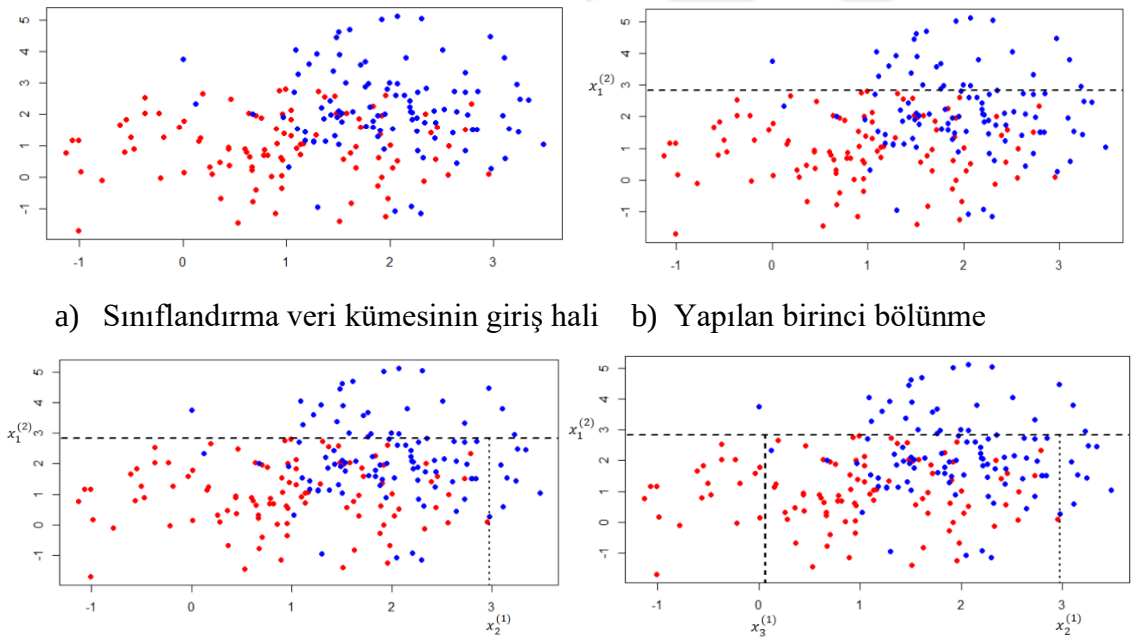
Şekil 5.6. Karar ağacı görüntüsü (URL-18)

Tez çalışmasında Karar ağaçlarının CART Algoritma türünü kullanılmaktadır. Sınıflandırma ve Regresyon Ağacı’nın kısaltması olan CART, ilk olarak 1984 yılında Briman ve arkadaşları tarafından yayınlanan ikili (binary) bir karar ağacı algoritmasıdır. Şekil 5.7’de CART algoritma türü için bir örnek gösterilmiştir.

CART algoritmasında, her karar ağacının düğümünün iki torunu vardır. Ağacın her bir adımında, düğümde oluşturulan kural, verilen örnek kümesini (eğitim örneği) iki parçaya böler - kuralın uygulandığı kısım (torun - sağ) ve kuralın uygulanmadığı kısım (torun - sol). Hangi düğümün karar veya kök düğüm olacağına biz karar vermiyoruz. Bu kararı almak için çeşitli değerler vardır. Bunlardan biri denklem (5.2)'de verilen Gini değeridir. Bu değer, alt kümenin saflık değerini verir.

$$Gini = 1 - \sum_j p_j^2 \tag{5.2}$$

p_j - j sınıfı gerçekleşme olasılığı. Her sınıf için hesaplanır ve sonuçların karelerinin toplamı birden çıkarılır. Gini değeri 0 ile 1 arasında bir değer alır ve sonuç 0'a ne kadar yakınsa ayrım o kadar iyi olur.



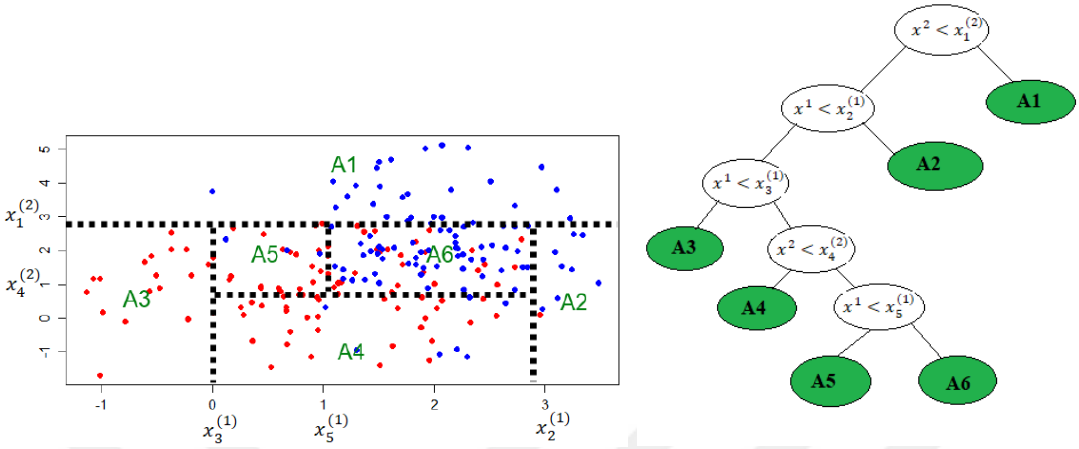
a) Sınıflandırma veri kümesinin giriş hali b) Yapılan birinci bölünme

c) Yapılan ikinci bölünme d) Yapılan üçüncü bölünme

Şekil 5.7. CART algoritmasında bölünme prosedürü (URL-18)

Algoritmanın durdurulması aşağıdaki durumlarda gerçekleşir:

- Düğümün maksimum derinliğine ulaşıldığı zaman
- Alt kümedeki öğe sayısı belirli bir eşikten az olduğunda



a) Yapılan sonuncu bölünme b) CART algoritmasının çalışma şeması

Şekil 5.8. CART algoritmasında bölünme prosedürünün sonucu (URL-18)

5.1.4. Rastgele Orman Sınıflandırıcısı (Random Forest, RF)

Rastgele Orman sınıflandırıcısı, ağaçlar arasındaki korelasyonu ortadan kaldıran karar ağaçlarında küçük bir iyileştirmedir (Stamp, 2017). Kontrollü öğrenme algoritması olarak bilinen RF ilk olarak, eğitim setlerinden birçok karar ağacı oluşturur. Diğer bir deyişle, bir orman yaratır ve rastgelelik sağlar. İstikrarlı bir tahmin elde etmek için bu karar ağaçları birleştirilir. Bu ağaçların inşası sırasında, her bölme (split) gerçekleştirilmeden önce dikkate alınacak p tahmin edicilerinden yalnızca t tanesi rastgele seçilir. Her bölme için, yeni bir t öngörücü seti oluşturulur. $t \approx \sqrt{p}$, her bölmede dikkate alınan öngörücülerin sayısıdır.

Rastegele Orman Sınıflandırıcısının önemli parametrelerinden biri ağaç sayısı (n_estimators) parametresidir. Çoğu durumda, ne kadar çok ağaç olursa kalite o kadar iyi olur, ancak RF kurulum ve çalışma süreleri de orantılı olarak artar.

Rastgele Orman Sınıflandırıcısın avantajları:

- Gradyan artırma sonuçlarıyla karşılaştırılabilir, yüksek tahmin doğruluğuna sahiptir
- Detaylı parametre ayarlamaları gerektirmez, varsayılan ayarlarla iyi çalışır
- Öznitelik değerlerinin ölçeklendirilmesine ve diğer monoton dönüşümlere karşı duyarsızdır

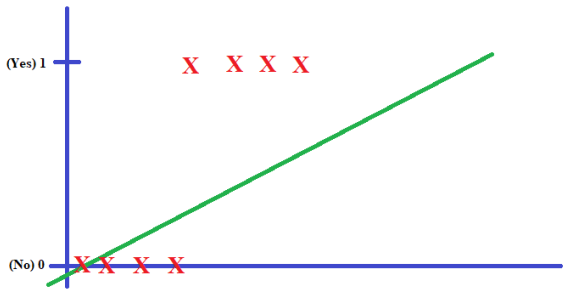
- Veri kümesinde eksik olan verilerle iyi çalışır

Rastgele Orman Sınıflandırıcısın dezavantajları:

- Modellerin büyük boyutlu olması
- Rastgele bir ağaç algoritması uygulamak için büyük ölçüde bilgi işlem kaynağı gerekir
- Değişkenlerin önemini değerlendirmek için kullanılan p-değerleri gibi formel çıkarımlar yoktur

5.1.5. Lojistik Regresyon (Logistic Regression, LR)

İkili sınıflandırma problemleri üzerinde çalışabilen istatistiksel bir modele dayalı bir sınıflandırma algoritmasıdır ve denetimli makine öğrenmesi yöntemlerinden biridir (Stamp, 2017). Algoritmanın adı regresyon kelimesinden gelmesine rağmen, algoritma bir sınıflandırma problemine dayanmaktadır. Lojistik regresyonu kullanılmasının nedeni ikili sınıflandırma tahminleri için doğrusal regresyon kullanılmamasıdır. Şekil 5.9'da verildiği gibi, Sonuç özniteliği 0 ve 1 değerine sahipse, bu veri kümesinde düz bir çizgi çizmeye çalışıldığı zaman, basitçe uymayacaktır.



Şekil 5.9. Doğrusal regresyonun ikili sınıflandırma için uyumsuzluğu (Stamp, 2017)

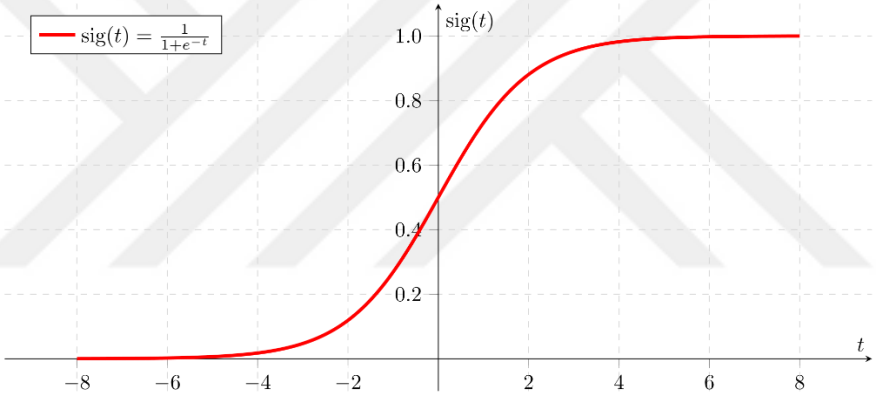
Bu nedenle lojistik regresyon modeli kullanışlıdır. En iyi uyum çizgisine doğru bir eğimi vardır, bu da onu nitel (kategori) verileri tahmin etmek için çok daha uygun hale getirir.

Lojistik regresyonun bir bükülmeye sahip olmasının nedeni, onu hesaplamak için doğrusal bir denklem kullanılmamasıdır. Bunun yerine, bir lojistik regresyon modeli sigmoid (lojistik

regresyonda kullanıldığı için lojistik fonksiyon olarak da adlandırılır) kullanımına dayanır. Şekil 5.11'de gösterilen Sigmoid fonksiyonu, denklem (5.3) ile ifade edilir.

$$y = \frac{1}{1+e^{-z}} \tag{5.3}$$

Denklem (5.3), girdi parametresi olarak gösterilen z'yi kabul eder ve öngörücülerin doğrusal kombinasyonunu temsil eder ve sınıfın olasılığını hesaplayan çıktı parametresi y'yi hesaplar. Çıktı değeri 0 ile 1 arasında bulunur, 0, sınıfın sıfır olasılığını temsil ederken, 1, sınıfın değişmez olasılığını temsil eder.



Şekil 5.10. Sigmoid fonksiyonu (URL-19)

Sigmoid fonksiyonunun birçok avantajı vardır. Bu avantajlarından en önemlisi sayısal bir değer alıp o değer için (0, 1) aralığında bir değer üretir. Tahminler için, lojistik regresyonu kullanmak için, genellikle tam olarak bir kesme noktası tanımlaması gerekir. Bu kesme noktası tipik olarak 0,5'tir eşittir. Örnek olarak LR algoritması 0.5 değerinin altında bir değer verirse bu dosyanın normal olduğunu gösterir. Diğer durumda, alınan değer 0.5 üzerinde olursa bu dosyanın kötücül olduğunu gösterir.

6. ARAÇLAR VE YÖNTEMLER

Bu bölümde, kötücül yazılım tespitinde kullanılacak veri kümesinin nasıl oluşturulduğu açıklanmaktadır. Veri analizi için, detaylı olarak yukarıda bahsedilen beş farklı makine öğrenimi algoritması kullanılmaktadır.

6.1. Veri Kümesi

Bilindiği gibi, makine öğreniminde analiz için bir veri kümesine ihtiyaç vardır. Bir veri kümesinin elde edilmesinin birkaç yolu vardır. Bazıları bu işlemi manuel olarak yapar - daha uzun sürer, ancak hata oranı az olur. Bazıları ise verileri otomatik toplar, genellikle özel fonksiyonlar kullanılarak yapılır.

Bu tez çalışmasında veri kümesi manuel olarak oluşturulmuştur. Veri kümesi, kötü amaçlı ve normal veriler olmak üzere, 3.500 veriden oluşmaktadır. Toplanılan verilerin yarısı normal (benign), diğer yarısı kötü amaçlı (malware) dosyalardan oluşmuştur. Normal dosyalar (exe ve dll) Windows 10 işletim sisteminin system 32 klasöründen alınmıştır. Kötü amaçlı dosyalar İnternetten indirilmiştir. Veri kümesi toplam 6 öznitelikten (features) oluşmaktadır. Tablo 6.1'de bu özniteliklerin genel açıklaması yer almaktadır.

Tablo 6.1. Eğitim setinde kullanılan öznitelikler

Parametre ismi(Feature name)	Açıklama ve örnekler
Hash	Hash işlevi, bilginin belirli bir uzunluktaki kısa bir diziye matematiksel olarak dönüştürülmesidir.
Virtual size	Belleğe yüklendiğinde bölümün toplam boyutu.
Raw size	Raw size (ham boyut), dosya bölümünün dosyadaki verilerinin tam boyutudur.
File size	Dosyanın başlangıç boyutu.
Entropy	Entropi, dosyalardaki bir ikili dizinin rasgele dağılımının ve tahmin edilebilirliğinin bir göstergesidir.
Number of sections	Bölüm sayısı. Başlıkları hemen takip eden bölüm tablosunun boyutunu belirtir.

Bu özellikleri toplamak için üçüncü bölümde açıklanan PeStudio yazılımını kullanılmıştır. Özellikler topladıktan sonra veri kümesi son olarak bir csv dosyası formatında oluşturuldu. Bu veri kümesinin analiz için hazır hale geldiğini anlamına gelir.

6.2. Analiz Aşaması

Verileri analiz etmek için farklı programlar ve ortamlar vardır. Bu tezde Anaconda Navigator geliştirme ortamı tarafından sağlanan Jupyter Notebook kullanıldı.

Jupyter Notebook'u çalıştırdıktan sonra yapılması gereken ilk şey, veri kümesinin analizinde faydalı olacak Numpy, Pandas, Seaborn, Matplotlib kütüphaneleri yüklemektir. Daha sonra, veri kümesi yüklenir. Veri kümesinin yüklenmesi ve içeriğine ait ilk beş satır Şekil 6.1'de sunulmaktadır.

	hash	virtual_size	raw_size	file_size	entropy	size_of_optional_header	num_sections	malware	classification
0	D0BE9EEE425ACECC5469286424A44405	45416	53248	232128	6.750	224	4	1	trojan
1	5E28AA7D4E48C41841852A991720E0C9	643192	634880	638976	5.900	224	6	1	trojan
2	DA7CB13D5A2E5271575D40A1F8A9FE6F	111842	111104	122920	5.241	240	6	0	benign
3	A1A3F9E8A096B629CBA87F9A1EF0CAC6	6057	6656	7680	4.478	224	4	0	benign
4	5037D8E6670EF1D89FB6AD435F12A9FD	22166	22016	23040	5.320	240	7	0	benign

Şekil 6.1. Veri kümesinin ilk beş satırı

Şekil 6.2'de özniteliklerle ilgili detaylı bilgi sunmaktadır. Burada her öznetilikte veri sayısı, özniteliğin tipi (Dtype), null olmayan değerler yer almaktadır. Örnek: hash – 3500- non-null-object.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3500 entries, 0 to 3499
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   hash                     3500 non-null   object
 1   virtual_size             3500 non-null   int64
 2   raw_size                 3500 non-null   int64
 3   file_size                3500 non-null   int64
 4   entropy                  3500 non-null   float64
 5   size_of_optional_header  3500 non-null   int64
 6   num_sections             3500 non-null   int64
 7   malware                  3500 non-null   int64
 8   classification           3500 non-null   object
dtypes: float64(1), int64(6), object(2)
memory usage: 246.2+ KB
```

Şekil 6.2. Veri kümesinde olan özniteliklerle ilgi bilgi

‘describe’ fonksiyonu veri kümesinde sayısal olan özniteliklerle bir seriyi tanımlar. Count indeksi her öznitelikte kaç tane veri sayısı girildiğini gösterir. Mean indeksi her özniteliğin ortalamasını gösterir. Std indeksi her özniteliğin dizi elemanlarının değerlerinin standart sapmasını hesaplar. Min indeksi her özniteliğin girilen minimal değerini gösterir. 25% indeksi, her öznitelik için girilen verilerin %25’den düşük bir değere veya aynı değere sahip olduğu anlamına gelir. Örnek olarak: entropy özniteliğinin 25% indeksinde aldığı değer 5.90 eşittir. O zaman bu öznitelikte olan değerlerin, 25% - 5.90 değerinden düşük veya o değere eşit bir değer olduklarını gösterir. 50% indeksi, her öznitelik için girilen verilerin %50’den düşük bir değere veya aynı değere sahip olduğu anlamına gelir. 75% indeksi, her öznitelik için girilen verilerin %75’den düşük bir değere veya aynı değere sahip olduğu anlamına gelir. Max indeksi her özniteliğin girilen maksimal değerini gösterir. Veri kümesine ait özet Şekil 6.3’te sunulmaktadır.

```
df.describe()
```

	virtual_size	raw_size	file_size	entropy	size_of_optional_header	num_sections	malware
count	3.500000e+03	3.500000e+03	3.500000e+03	3500.000000	3500.000000	3500.000000	3500.000000
mean	9.962137e+05	6.182228e+05	7.563703e+05	6.337554	232.155429	5.736000	0.500000
std	3.126842e+06	1.448170e+06	1.683473e+06	1.034231	7.999633	2.684627	0.500071
min	1.227000e+03	2.560000e+03	3.584000e+03	0.048000	224.000000	1.000000	0.000000
25%	1.167270e+05	8.947200e+04	1.154070e+05	5.904750	224.000000	4.000000	0.000000
50%	2.930580e+05	2.340225e+05	2.903040e+05	6.284000	240.000000	6.000000	0.500000
75%	7.749685e+05	6.225920e+05	7.620060e+05	6.947500	240.000000	7.000000	1.000000
max	6.188646e+07	3.231488e+07	3.283346e+07	8.000000	240.000000	73.000000	1.000000

Şekil 6.3. ‘describe’ fonksiyonunun veri kümesine uygulanması

Bu çalışmada, beş farklı kategoriden 1750 adet kötü amaçlı dosya kullanılmıştır. Şekil 6.4’te veri kümesinde kullanılan iyi huylu (benign) ve kötücül yazılmlar gösterilmiştir (kategori solucan, arka kapı, virüs, reklam yazılımları ve trojan içeriyor).

```
df['classification'].value_counts()
```

```
benign      1750
adware       350
virus        350
worm         350
trojan       350
backdoor     350
```

Şekil 6.4. Kötücül yazılım sınıflarının dağılımı

Analize geçmeden önce son yapılması gereken işlem, veri ölçeklendirme işlemidir. Elde edilen veriler çeşitli boyut ve ölçekleri içerir. Verilerin farklı ölçekleri veri kümesini modellenmeyi olumsuz yönde etkiler. Verileri aynı ölçeğe getirilmesi için StandartScaler fonksiyonu kullanılmıştır.

StandardScaler, girdi veri kümesinin işlevsellik aralığını standart hale getirmek için birçok makine öğrenimi modelinden önce bir ön işleme adımı olarak gerçekleştirilen önemli bir tekniktir. Python'daki sklearn kütüphanesi, veri değerlerini standart bir formatta standartlaştırmak için StandardScaler() işlevini sunar.

Ölçülendirme işleminin aşamaları sunulmaktadır:

- Adım 1. StandardScaler işlevini kullanmak için, StandarScaler sklearn kütüphanesinden import edilmiştir
- Adım 2. StandardScaler() fonksiyonunun nesnesi tanımlanır
- Adım 3. Fit() fonksiyonunu veri kümesine uygulanıyor, aynı zamanda veri kümesinde kullanılmayan özentilikler çıkarılıyor
- Adım 4. scaler nesnesine transform() fonksiyonu uygulanıyor

Elde edilen ölçeklenmiş verileri X ve y değişkenlerine ayrıştırılmıştır. X giriş (input) değişkenlerine, y ise hedef (Target) değişkenine ayrıştırılıyor.

Daha sonra, train_test_split fonksiyonunu sklearn kütüphanesinden import edilmektedir. Veri kümesi burada train ve test olarak ikiye ayrılmaktadır. Test için ayrılan küme yüzdesi %30 (0.3)olarak belirlenmiştir.

Tüm bu işlemlerden sonra veri kümesi makine öğrenmesi algoritmalarının kullanılmasına uygun hale getirilmiştir.

6.3. Algoritmaların Başarısını Test Etme Ölçütleri

Bu tez çalışmasında, kötücül yazılım tespitinde Tablo 6.1'de verilen PE dosyalarından çıkarılan virtual_size, raw_size, file_size, entropy, size_of_optional_header öznitelikleri

kullanılmıştır. Algoritmaların başarım performansını değerlendirmek için literatürde en yaygın olan doğruluk, kesinlik, duyarlılık ve F1 skoru parametreleri kullanılmıştır. Her algoritma için, ilave bilgiler sağlamak üzere Şekil 6.2'de verilen bir karışıklık matrisi oluşturulur. Karışıklık Matrisi, sınıflandırma yapan bir Makine Öğrenimi (ML) modeli için bir performans ölçüsüdür.

	Gerçekleşen (Actual)	
Tahminlenen (Predicted)	**Doğru Pozitif (DP)**	**Yanlış Pozitif (YP)**
	Yanlış Neqatif (YN)	**Doğru Neqatif (DN)**

Şekil 6.5. Karışıklık Matrisi (URL-20)

Şekil 6.5'te DP ve DN algoritmanın doğru olarak tahminlediği, YP ve YN modelin yanlış olarak tahminlediği alanları göstermektedir. DP kötücül dosyaların doğru bir şekilde sınıflandırıldığı değeri, YP zararsız dosyaların kötücül olarak sınıflandırılma sayısını, DN kötücül olmayan dosyaların doğru sınıflandırma sayısını, YN kötücül dosyaların iyi kuylu olarak sınıflandırılma sayısını göstermektedir.

Doğruluk (accuracy): Bir algoritmanın doğruluğu, denklem (6.1)'de gösterildiği gibi gerçekleştirilen doğru tahmin sayısının toplam kötücül / kötücül olmayan dosya sayısına bölünmesiyle hesaplanır.

$$\text{Doğruluk} = \frac{(\text{DP}+\text{DN})}{(\text{DP}+\text{YP}+\text{DN}+\text{YN})} \tag{6.1}$$

Kesinlik (Precision): Pozitif (kötücül) olarak tahmin edilen değerlerin gerçekten kaç tanesinin kötücül olduğunu gösteren yüzdesel değerdir.

$$\text{Kesinlik} = \frac{\text{DP}}{\text{DP}+\text{YP}} \tag{6.2}$$

Duyarlılık (Recall): Kötücül olarak tahmin edilen tane gerçek sonucun doğru tahmin edildiği gösterir.

$$\text{Duyarlılık} = \frac{\text{DP}}{\text{DP+YN}} \quad (6.3)$$

F1-Skoru (F1-Score): Kesinlik ve duyarlılık değerlerinin harmonik ortalaması alınarak elde edilir.

$$\text{F1} = 2 * \frac{\text{kesinlik} * \text{duyarlılık}}{\text{kesinlik+duyarlılık}} \quad (6.4)$$

6.4. Öznitelik Seçimi

Öznitelik seçimi, veri kümesinde olan en yararlı öznitelikleri bulma sürecidir. Veri kümesinin eğitiminden önce özellik seçimi için başka önemli nedenler de vardır. Birincisi, özellik uzayının boyutluluğunu azaltır ve modelin eğitim süresini kısaltır (Sura ve diğ., 2021). Öznitelik seçim yöntemleri üç gruba ayrılır: filtre yöntemleri (filters), sarmalayıcı yöntemler (wrappers) ve gömülü yöntemler (embedded). Aşağıda üç grup özellik seçim yöntemi bulunmaktadır: filtreler (filters), sarmalayıcılar (wrappers) ve gömülü yöntemler (embedded).

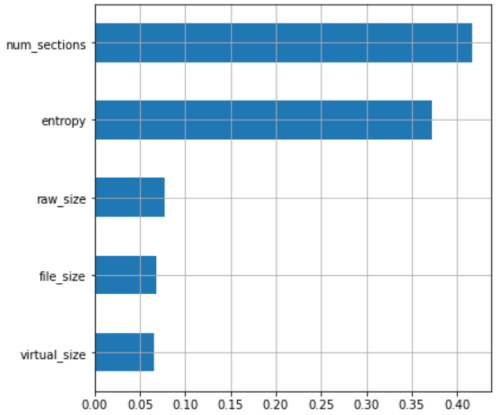
Filtreleme yöntemleri: Özelliklerin önemini hesaplamak için bir özellik ile hedef değişken arasındaki ilişkiyi dikkate alan yöntemlerdir. Yapılan işlemler sonucunda veri kümesi filtrelenmiş ve ilgili öznitelikler seçilerek bir alt küme oluşturulmuş olur. En sık kullanılan yöntemler Pearson korelasyonu ve ki-kare yöntemidir.

Sarmalayıcı yöntemler, özelliklerin bir alt kümesi ile modeller oluşturur ve modelin etkinliğini ölçer. Bunun için iki tür arama stratejisi vardır;

İleri arama (Forward selection): boş bir özellik kümesiyle başlar ve ardından model kalitesinde en iyi kazancı sağlayan özellikleri aşamalı olarak ekler.

Geriye doğru arama (Backward selection): tüm özelliklerden oluşan bir küme ile başlar, ardından her iterasyonda "en kötü" özelliği kaldırır.

Gömülü yöntemler (Embedded): Bu öznitelik seçimi yöntemi, Makine Öğrenimi modellerinin sağladığı iç görülerle de elde edilebilir. Örneğin, bu tez çalışmasında Rf algoritmasının kendine ait öznitelik seçim metodu kullanılmaktadır.

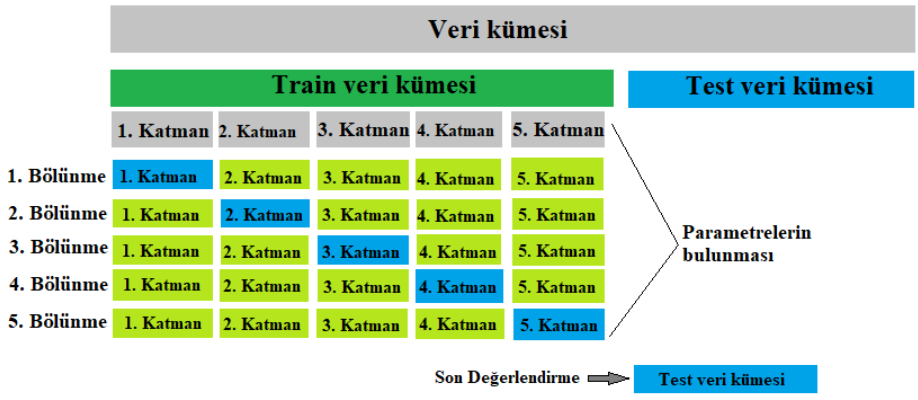


Şekil 6.6. Rf yöntemi ile öznitelikler seçimi için alınan sonuç

Şekil 6.6'ya bakıldığında num_sections özniteliğinin daha ağırlıklı bir parametre olduğu görülmektedir. Deneyimler ve sonuç bölümünde num_sections ve entropy özniteliklerini kullanarak alınan sonuçlar sunulmaktadır.

6.5. K Katlamalı Çapraz Doğrulama

K-fold cross validation (k katlamalı Çapraz doğrulama), makine öğrenme modellerinin başarılarının değerlendirilmesi için literatürde kullanılan bir yöntemdir. Bu yöntemde, veri kümesi eğitim ve test seti olarak ayrılır ve bu işlem için seçilen yöntem modelin başarısını önemli ölçüde etkileyebilir. Şekil 6.7'de K Katmanlı Çapraz doğrulama örneği verilmektedir (k = 5).



Şekil 6.7. K Katmanlı Çapraz doğrulama (URL-21)

7. DENEYİMLER VE SONUÇ

Bu bölümde her algoritma için elde edilen performans sonuçları verilmektedir.

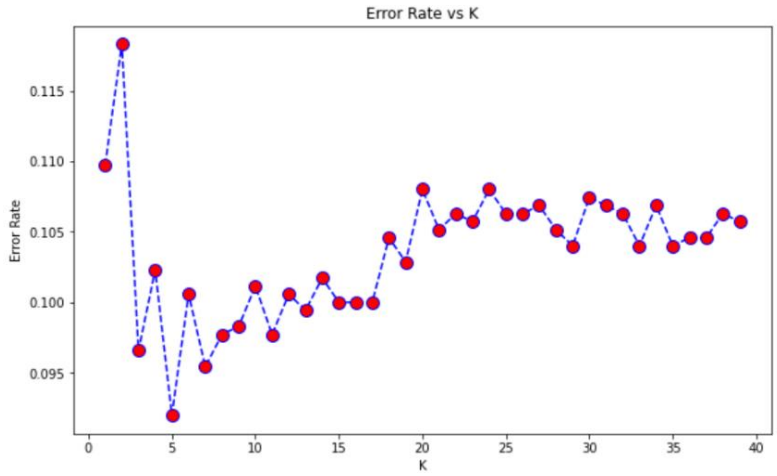
7.1. K - en Yakın Komşular (K-Nearest Neighbors, KNN) Sonuçları

KNN algoritmasını uygulamak için sklearn.neighbors.KneighborsClassifier fonksiyonu kullanılmıştır. **K** parametresi n_neighbors değeri ile temsil edilir ve 5 olarak ayarlanmıştır.

KNN algoritması için kullanılan parametreler:

- n_neighbors=5
- weights='uniform'
- leaf_size=30
- p=2
- metric='minkowski'

Bu veri kümesi için, error_rate yöntemi ile 5 değeri en uygun değer olarak hesaplanmıştır.



Şekil 7.1. **K** değerinin hata oranı

Şekil 7.1'de seçilen farklı **K** değerleri için hata oranları gösterilmektedir. Deneysel çalışma sonucunda KNN algoritmasının doğruluk oranı (ACC) 0.91, kesinlik değeri 0.91, duyarlılık değeri 0.91 ve F1-skoru 0.91 olarak elde edilmiştir.

7.2. Destek Vektör Makinesi (Support Vector Machine, SVM) Sonuçları

SVM algoritmasını uygulamak için sklearn.svm SVC fonksiyonu kullanılmıştır. Bu çalışmada regülarizasyon parametresi *C* 100 değerine set edilirken, Kernel parametresi ise 'RBF' yöntemi olarak seçilmiştir.

SVM algoritması için kullanılan parametreler:

- C=100
- kernel='rbf'
- degree=3
- gamma='scale'

Deneysel çalışma sonucunda SVM algoritmasının doğruluk oranı (ACC) 0.90, kesinlik değeri 0.91, duyarlılık değeri 0.90 ve F1-skoru 0.90, olarak elde edilmiştir.

7.3. Karar Ağaçları (Desicion Tree, DT) Sonuçları

Karar Ağaçları algoritmasını uygulamak için sklearn.tree DecisionTreeClassifier fonksiyonu kullanılmıştır. Bu çalışmada, DT algoritma türlerinden Sınıflandırma ve Regresyon Ağacı (Classification And Regression Tree, CART) algoritması kullanılmıştır. max_depth parametresi 5 olarak alınmıştır. Karar ağaçları için kullanılan parametreler:

- max_depth=5
- criterion='gini'
- splitter='best'

Deneysel çalışma sonucunda DT algoritmasının doğruluk oranı (ACC) 0.91, kesinlik değeri 0.91, duyarlılık değeri 0.91 ve F1-skoru 0.91 olarak elde edilmiştir.

7.4. Rastgele Orman Sınıflandırıcısı (Random Forest, RF) Sonuçları

Rastgele Orman Sınıflandırıcısı algoritmasını uygulamak için sklearn. ensemble RandomForestClassifier fonksiyonu kullanılmıştır. Bu çalışmada, n_estimators parametresi

50 olarak, diğer parametreler ise varsayılan olarak alınmıştırlar. Deneysel çalışma sonucunda RF algoritmasının doğruluk oranı (ACC) 0.92, keskinlik değeri 0.91, duyarlılık değeri 0.93 ve F1-skoru 0.92 olarak elde edilmiştir.

Rastegele Orman Sınıflandırıcısı algoritması için kullanılan parametreler:

- n_estimators=100
- criterion='gini'
- max_depth=None
- min_samples_split=2

7.5. Lojistik Regresyon (Logistic Regression, LR) Sonuçları

Bu çalışmada Lojistik Regresyon algoritması için kullanılan parametreler:

- Cs=10
- fit_intercept=True
- penalty='l2'
- solver='lbfgs'
- max_iter=100

Deneysel çalışma sonucunda LR algoritmasının doğruluk oranı (ACC) 0.85, kesinlik değeri 0.86, duyarlılık değeri 0.85, F1-skoru 0.85 olarak elde edilmiştir.

Tablo 7.1'deki sonuçlardan da görüldüğü gibi en yüksek doğruluk başarı oranı %92 ile Rastgele Orman Sınıflandırıcısına ait olduğu görülmektedir.

Tablo 7.1. Tüm algoritmalar için alınan sonuçlar

Algoritma türü	Doğruluk	Kesinlik	Duyarlılık	F1-skoru
K-en Yakın Komşular	0.91	0.91	0.91	0.91
Karar Ağaçları	0.91	0.91	0.91	0.91
Rastgele Orman	**0.92**	**0.91**	**0.93**	**0.92**
Destek Vektör Makinesi	0.90	0.91	0.90	0.90
Lojistik Regresyon	0.85	0.86	0.85	0.85

Öznitelik seçimi sonrası alınan sonuçlar tablo 7.2'de sunulmaktadır. Analiz için veri kümesinden RF öznitelik yöntemi ile seçilen iki ağırlıklı öznitelik üzerinden (entropy ve num_sections) test edilmiştir.

Tablo 7.2. Öznitelik seçimi sonrası tüm algoritmalar için alınan sonuçlar

Algoritma türü	**Doğruluk**	**Kesinlik**	**Duyarlılık**	**F1-skoru**
K-en Yakın Komşular	0.92	0.92	0.92	0.92
Karar Ağaçları	**0.93**	**0.93**	**0.93**	**0.93**
Rastgele Orman	0.91	0.91	0.91	0.91
Destek Vektör Makinesi	**0.93**	**0.93**	**0.93**	**0.93**
Lojistik Regresyon	0.85	0.86	0.85	0.85

Öznitelik seçimi yapıldığı zaman daha iyi sonuçlar alınmıştır. En yüksek doğruluk başarı oranı %93 ile K-en Yakın Komşular ve Destek Vektör Makinesine ait olduğu görülmektedir.

K Katmanlı Çapraz doğrulama sonuçları tablo 7.3'de sunulmaktadır. K değeri, K=5 olarak alınmıştır. Alınan sonuçlara bakıldığı zaman, K ortalaması Rastgele Orman Sınıflandırıcısında yüksek olduğu görülmektedir.

Tablo 7.3. K Katmanlı Çapraz doğrulama sonuçları

Algoritma türü	**K – 1**	**K-2**	**K-3**	**K-4**	**K-5**	**K-ortalama**
K-en Yakın Komşular	0.91	0.90	0.91	0.92	0.92	0.91
Karar Ağaçları	0.92	0.89	0.90	0.93	0.93	0.92
Rastgele Orman	**0.92**	**0.9**	**0.96**	**0.96**	**0.92**	**0.93**
Destek Vektör Makinesi	0.91	0.88	0.93	0.90	0.91	0.91
Lojistik Regresyon	0.87	0.85	0.82	0.79	0.87	0.84

8. SONUÇ

Bu tez çalışmasında oluşturulan 5 statik özellik kullanılarak kötücül yazılım tespiti için Karar Ağaçları, K-En yakın Komşular, Rastgele Orman Sınıflandırıcısı, Destek Vektör Makinesi (SVM) ve Lojistik Regresyon yöntemleri kullanılmıştır. Yöntemler arka kapı, virüs, truva atı, reklam yazılımları ve solucan olmak üzere beş tür kötücül yazılım üzerinde değerlendirildi: Sonuçlar, rastgele orman sınıflandırıcısının tüm kötücül yazılım türlerinde diğer dört algoritmadan daha iyi performans sergilediğini gösterdi. Bunun nedeni, rastgele ormanların küçük verilerin analizi için daha uygun olmasıdır. Öznitelik seçimi yapıldığı zaman ise K-en Yakın Komşular ve Destek Vektör Makineleri algoritmalarının daha başarılı olduklarını görülmektedir. Bunun nedeni, daha az sayıda öznitelikler ile K-en Yakın Komşular ve Destek Vektör Makineleri algoritmalarının daha iyi çalışabilmeleridir.

Gelecekteki çalışmalarda, daha fazla veri kullanılmasının yanı sıra derin öğrenme mimarileri kullanılarak kötücül çalıştırılabilir programların tespit edilmesi performansını artıracak yeni özniteliklerin bulunması planlanmaktadır.

KAYNAKLAR

Abdessadki, I., Lazaar, S. (2019). New classification based model for malicious PE files detection. *Int. J. Comput. Netw. Inform. Secur. 11*(6), 43-46.

Abhishek, K., Kumar, A., Shah, K., Patel, D., Jain, Y., Chheda, H., Nerurkar, P. (2020). Malware Detection Using Machine Learning. *KGSWC 2020: Knowledge Graphs and Semantic Web,* 99. DOI:10.1109/ACCESS.2021.3094132

Alazab, M., Vinayakumar, R., Soman, K.P., Venkatraman, S., Poornachandran, P. (2019). Intelligent Malware Detection Using Deep Learning. *IEEE Access, 7*, 46717 – 46738. DOI: 10.1109/ACCESS.2019.2906934.

Alsmadi, T., Alqudah, N. (2021). A Survey on malware detection techniques, *2021 International Conference on Information Technology (ICIT),* 371-376. DOI: 10.1109/ICIT52682.2021.9491765.

Anderson, B., Quist, D., Neil, J., Lane, T., Storlie, C.(2011). Graph-based malware detection using dynamic analysis. *J. Comput. Virol. 7(4),* 247–258.

Aslan, O., Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches, Digital Object Identifier 10, *IEEE Access (8),* 6249 – 6271.

Austin, T., Mehne, B., Reetuparna, D., William, A. (2015). Getting in control of your control flow with control-data isolation, *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO),* 79-90.

Baydogmus, G.K., Emanet, S., Demir, O. (2021). Effects of Feature Selection Methods on Machine Learning Based Intrusion Detection System Performance, *DUJE (Dicle University Journal of Engineering), 12*(5),743-755.

Belaoued, M., Mazouzi, S. (2015). A Real-Time PE-Malware Detection System Based on CHI-Square Test and PE-File Features. In: Amine, A., Bellatreche, L., Elberrichi, Z., Neuhold, E., Wrembel, R. (eds) Computer Science and Its Applications. CIIA 2015. *IFIP Advances in Information and Communication Technology,* 456. Springer, Cham. DOI:10.1007/978-3-319-19578-0_34

Bowles, S., Hernandez-Castro, J. (2015). The first 10 years of the Trojan Horse defence. *Computer Fraud & Security, 2*(1), 5-13. DOI:10.1016/S1361-3723(15)700059

Cepeda C., Dan Lo, C.T., Ordonez P. (2015). Feature Selection and Improving Classification Performance for Malware Detection. *IEEE International Conferences on Big Data and Cloud Computing (BDCloud) Social Computing and Networking (SocialCom) Sustainable Computing and Communications (SustainCom),* 560-566.

Chhabra, D. K. R. (2014). Feature selection and clustering for malicious and benign software characterization, Master Thesis, Master of Science in Computer Science Information Assurance, University of New Orleans.

Christian, C. (2011). Estonia after the 2007 Cyber Attacks: Legal, Strategic and Organisational Changes in Cyber Security. Estonia after the 2007 Cyber Attacks: Legal, Strategic and Organisational Changes in Cyber Security, *The Journal of Cyber Warfare and Terrorism, 1, 11.*

Currie, M., Fidler, B. (2015). The production and interpretation of ARPANET maps. *IEEE Annals of the History of Computing 37(1),* 44–55, 10.1109.

Denil, M., Matheson, D., De Freitas, N. (2014). Narrowing the Gap: Random Forests In Theory and In Practice, *International Conference on Machine Learning (ICML),* 23-44.

Dolu, E., Ecemiş, İ.N., Keskin, A. (2019). *Mirai botnet,* 2.

Ehrenfeld, J. M. (2017). Wannacry, cybersecurity and health information technology: A time to act. Journal of medical systems, 41, 104.

Elovici, Y., Shabtai, A., Moskovitch, R., Tahan, G., Glezer, C. (2007). Applying machine learning techniques for detection of malicious code in network traffic. *In Proceedings of the 30th Annual German Conference on Advances in Artificial Intelligence, KI '07,* Berlin, Heidelberg, 978-3-540-74564-8, 44–50.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd Ed.). O'Reilly.

Hemant, R., Swati, A., Sanjay K.S., Mohit, S. (2018) .Malware Detection Using Machine Learning and Deep Learning. *International Conference on Big Data Analytics, BDA 2018: Big Data Analytics,* 402–411.

Ijaz, M., Durad, M.H., Ismail, M. (2019). Static and Dynamic Malware Analysis Using Machine Learning. *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST),* 118-124.

Kucharska, E., (2020). SS symmetry. *Mdpi,* 1–19.

Kumar, A., Kuppusamy, K.S., Aghila, G. (2019). A learning model to detect maliciousness of portable executable using integrated feature set, *Journal of King Saud University - Computer and Information Sciences, 31,* 252-265.

Kushner, D. (2013). "The real story of stuxnet," *in IEEE Spectrum, 50*(3), 48-53. DOI: 10.1109/MSPEC.2013.6471059.

Liang, G., Dai, C., Pang, J. (2016). A behavior-based malware variant classification technique. *Journal of Computer Virology and Hacking Techniques, 12*(2). DOI:10.1007/s11416-015-0244-0

Mosli, R., Li, R., Yuan, B., Pan, Y. (2017). A Behavior-Based Approach for Malware Detection. *IFIP International Conference on Digital Forensics, 14*(3), 187–201.

Namanya, A.P., Awan, I.U., Disso, J.P., Younas, M. (2019). Similarity hash-based scoring of portable executable files for efficient malware detection in IoT. *Future Generation Computer Systems, 110,* 824-832.

O'Hanley, R. (2015). Spyware, Adware, Malware — It's All Sleazeware to Me. Information Systems Security. 13. 2-4. DOI: 10.1201/1086/44954

Perdisci, R., Vadrevu, P.,(2016). Scaling malware execution with sequential multi-hypothesis testing. *ASIA CCS '16: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security,* 771–782. DOI:10.1145/2897845.2897873

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). ''Scikit-learn: Machine Learning in Python,'' *Journal of Machine Learning Research*, 12, 2825 - 2830.

Pietrek, M. (1994). Peering inside the PE: A Tour of the Win32 Portable Executable File Format, Microsoft Systems, 15-38.

Schultz, M.G., Eskin, E. (2001). Data mining methods for detection of new malicious executables. In Security and Privacy, 2001. S&P 2001. Proceedings. *2001 IEEE Symposium on, IEEE*, 38–49.

Shafiq, M.Z., Tabish, S.M., Mirza, F., Farooq, M. (2009). PE-Miner: Mining Structural Information to Detect Malicious Executables in Real time, Recent Advances in Intrusion Detection. *Lecture Notes in Computer Science,* 5758, 121-141. DOI:10.1007/978-3-642-04342-0_7

Shukla, H., Patil, S., Solanki, D., Singh, L., Swarnkar, M., Thakkar, H.K. (2019). On the Design of Supervised Binary Classifiers for Malware Detection Using Portable Executable Files. *Proc. IEEE 9th Int. Conf. Adv. Comput. IACC 2019*, Tiruchirappalli, India, 41-146. DOI: 10.1109/IACC48062.2019.8971519

Shijo, P.V., Salim, A. (2015). Integrated Static and Dynamic Analysis for Malware Detection, Procedia Computer Science, *46*(2), 804-811.

Sihwail, R., Omar, K., Akram, K., Ariffin, Z. (2021). An Effective Memory Analysis for Malware Detection and Classification An Effective Memory Analysis for Malware, *Computers, Materials & Continua 2021*, *67*(2), 2301-2320. DOI:10.32604/cmc.2021.014510

Sikorski, M., Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software* (3rd ed.). No Starch Press.

Stamp, M. (2017). *Introduction to Machine Learning with Applications in Information Security* (2nd ed.). CRC Press.

URL-1: https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx (Erişim: Aralık 2022).

URL-2: https://www.thejuggernaut.com/pakistan-brain-computer-virus (Erişim: Aralık 2022).

URL-3: https://www.kaspersky.com.tr/about/press-releases/2022_turkiyede-arka-kapi - bilgisayar-kotu-amacli-yazilimlari-tespitleri-2022nin-ikinci-ceyreginde-29-azalsa-da-hala-yuksek-seviyede (Erişim: Aralık 2022).

URL-4: https://www.kaspersky.com.tr/about/press-releases/2022_meta-bolgesindeki - kullanicilarin-yarisindan-fazlasi-dijital-odemelerde-kimlik-avi-dolandiriciligiyla-karsilasti (Erişim: Aralık 2022).

URL-5: https://www.kaspersky.com.tr/about/press-releases/2022_turkiyede-casus - yazilimlardan-etkilenen-kullanici-sayisi-2022nin-ikinci-ceyreginde-5-artti (Erişim: Aralık 2022).

URL-6: https://www.kaspersky.com.tr/about/press-releases/2022_turkiyede-microsoft - office-aciklari-uzerinden-saldiriya-ugrayan-kullanici-sayisi-son-ceyrekte-17-artti (Erişim: Aralık 2022).

URL-7: https://www.kaspersky.com.tr/about/press-releases/2022_turkiyede-arka-kapi - bilgisayar-kotu-amacli-yazilimlari-tespitleri-2022nin-ikinci-ceyreginde-29-azalsa-da - hala-yuksek-seviyede (Erişim: Aralık 2022).

URL-8: https://www.kaspersky.com.tr/about/press-releases/2022_turkiyedeki-suistimal - tespitleri-2022nin-ikinci-ceyreginde-20-artti (Erişim: Aralık 2022).

URL-9: https://www.kaspersky.com.tr/about/press-releases/2022_turkiyede-kimlik-avi-ve - dolandiricilik-girisimleri-2022nin-ikinci-ceyreginde-79-artisla-tavan-yapti (Erişim: Aralık 2022).

URL-10: https://www.kaspersky.com.tr/about/press-releases/2022_meta-bolgesindeki-kullanicilarin-yarisindan-fazlasi-dijital-odemelerde-kimlik-avi-dolandiriciligiyla-karsilasti (Erişim: Aralık 2022).

URL-11: https://www.udemy.com/course/malware-analysis-d/learn/lecture/16798428#overview (Erişim: Ağustos 2022).

URL-12: https://www.pandasecurity.com/en/mediacenter/panda-security/30-years-cyberattacks-barrotes-wannacry/ (Erişim: Aralık 2022).

URL-13: https://news.drweb.ru/show/review/?lng=en&i=14582 (Erişim: Aralık 2022).

URL-14: https://www.researchgate.net/figure/Portable-executable-file-format_fig6_338355873 (Erişim: Ocak 2023).

URL-15: https://learn.microsoft.com/ru-ru/microsoft365/security/intelligence/worms-malware?view=o365-worldwide (Erişim: Aralık 2022).

URL-16: T. S. Körting, ‘‘How kNN (K-nearest neighbors) algorithm works,’’ https://www.youtube.com/watch?v=UqYde-LULfs, 2014 (Erişim: Ağustos 2022).

URL-17: T. S. Körting, ‘‘How SVM (Support Vector Machine) algorithm works,’’ https://www.youtube.com/watch?v=1NxnPkZM9bc&t=151s, 2014 (Erişim: Ağustos 2022).

URL-18: https://basegroup.ru/community/articles/math-cart-part1. (Erişim: Kasım 2022).

URL-19: https://machinelearningmastery.ru/derivative-of-the-sigmoid-function-536880cf918e/ (Erişim: Aralık 2022).

URL-20: https://www.kaggle.com/code/onurakkse/veri-bilimi-notlar (Erişim: Aralık 2022).

URL-21:https://scikit-learn.org/stable/modules/cross_validation.html (Erişim: Aralık 2022).

Urooj, U., Bander A.S., Al-rimy, A., Fuad A., Murad A.R. (2021) . Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. *Appl. Sci. 2022,* 12(1), 172.

Witten, D., James, G., Hastie, T., Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R (2nd ed.).

Yousaf, M.S., Durad, M. H., Ismail, M. (2019). Implementation of Portable Executable File Analysis Framework (PEFAF). *2019 16th Int. Bhurban Conf. Appl. Sci. Technol,* Islamabad, Pakistan, 671-675. DOI: 10.1109/IBCAST.2019.8667202

KİŞİSEL YAYIN VE ESERLER

Mukhtarov K., Yiğit H. Makine öğrenmesi yöntemleri ile kötücül yazılımın PE dosyaları üzerinden tespiti. *9 Uluslararası Marmara Fen ve Sosyal Bilimler Kongresi(IMASCON 2022)*, Kocaeli, 09-10 Aralık 2022.

ÖZGEÇMİŞ

İlk, orta ve lise öğrenimini Azerbaycan, Bakü'de tamamladı. 2014 yılında kazandığı National Aviation Academy'nin Havacılık ve Uzay Fakültesi Bilişim Sistemleri Mühendisliği Bölümü'nden 2018 yılında mezun oldu. 2019 yılında, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Bilişim Sistemleri Mühendisliği Anabilim Dalı'nda Yüksek Lisans eğitimine başladı. 2022 yılından beri Azerbaycan'da özel sektörde Veri Analisti olarak çalışmakta.