

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

**MAKİNE ÖĞRENMESİ KULLANARAK WINDOWS OLAY KAYITLARI
İLE DİNAMİK DAVRANIŞ ANALİZİ**

YÜKSEK LİSANS TEZİ

Göksun ÖNAL

Bilgi Güvenliği Anabilim Dalı

Tez Danışmanı: Prof.Dr. Ali Aydın SELÇUK

NİSAN 2025





TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Göksun ÖNAL



ÖZET

Yüksek Lisans Tezi

MAKİNE ÖĞRENMESİ KULLANARAK WINDOWS OLAY KAYITLARI İLE DİNAMİK DAVRANIŞ ANALİZİ

Göksun ÖNAL

TOBB Ekonomi ve Teknoloji Üniversitesi

Fen Bilimleri Enstitüsü

Bilgi Güvenliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Ali Aydın SELÇUK

Tarih: NİSAN 2025

Teknolojik gelişmelerin hız kazanmasıyla birlikte siber saldırılar ve zararlı yazılım tehditleri giderek karmaşılaşmakta ve daha tehlikeli hale gelmektedir. Bu durum, bilgi güvenliği alanında yenilikçi yaklaşımların ve etkili çözümlerin geliştirilmesi gerekliliğini ortaya koymaktadır. Bu tez çalışması, zararlı yazılım analizinde dinamik yöntemlerin etkinliğini artırmayı amaçlayarak, özellikle Windows işletim sistemine ait olay kayıt verilerinin detaylı incelenmesine odaklanmaktadır. Çalışmanın temel hedefi, izole edilmiş sanal ortamlarda zararlı ve zararsız yazılımların çalışma anında oluşturdukları işletim sistemi kayıt verilerinden anlamlı öznitelikler çıkarıp, makine öğrenmesi teknikleri kullanılarak zararlı yazılım tespitine yönelik yenilikçi bir yaklaşım geliştirmektir.

Bu amaç doğrultusunda, hem manuel hem de çevrimiçi doğrulama süreçleriyle zararlı veya zararsız olduğu belirlenen çalıştırılabilir dosyalar titizlikle toplanmıştır. Kontrollü sanal makine ortamlarında gerçekleştirilen deneysel uygulamalar sonucunda elde edilen veriler, öncelikle sayısal, bağlamsal, zaman tabanlı ve istatistiksel öznitelikler olarak sınıflandırılmıştır. Bu verilerden hangilerinin kullanılabilir olacağı belirlenirken, bazıları özellik mühendisliği teknikleriyle daha işlevsel hale getirilmiş ve model performansını artıracak kritik göstergeler oluşturulmuştur.

İlgili öznitelikler, Gradient Boosting, Logistic Regression ve Destek Vektör Makineleri gibi çeşitli sınıflandırma algoritmaları kullanılarak işlenmiş; elde edilen modellerin performansı ise K-Fold çapraz doğrulama yöntemi ile titizlikle değerlendirilmiştir. Uygulanan veri normalizasyonu ve özellik mühendisliği teknikleri, modellerin genel doğruluğunu artırırken, zararlı ile zararsız yazılımlar arasındaki ayrımı daha net ortaya koymuştur. Bu kapsamlı analiz, dinamik analiz temelli yaklaşımların, geleneksel statik yöntemlere kıyasla daha yüksek doğruluk oranları ve düşük yanılma payları sağladığını göstermiştir.

Ayrıca, bu çalışma literatürde sınırlı yer alan dinamik analiz yaklaşımlarına önemli katkılar sunmayı hedeflemektedir. Farklı veri setleri üzerinde gerçekleştirilen kapsamlı deneyler sonucunda, geliştirilen metodolojinin siber saldırılara karşı proaktif bir yaklaşım sağlayabileceği ortaya konulmuştur. Dinamik davranış analizi, hem eski hem de yeni zararlı yazılımların tespitinde geniş zaman aralıklarında etkili sonuçlar vermekte, böylece siber güvenlik stratejilerinin geliştirilmesinde kritik bir rol oynamaktadır. Sonuç olarak, bu tez çalışması, zararlı yazılım tespiti ve önlenmesinde dinamik analiz yöntemlerinin uygulanabilirliğini ve etkinliğini ortaya koyarak, siber güvenlik alanında yeni ufuklar açmaktadır.

Anahtar Kelimeler: Zararlı yazılım, Dinamik analiz, Windows olay kayıtları, Makine öğrenmesi, Sanal ortam, Özellik mühendisliği, Zararlı yazılım tespiti, Siber güvenlik

ABSTRACT

Master of Science

Dynamic Behavior Analysis through Novel Windows Event Logs with Machine
Learning

Göksun ÖNAL

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Ali Aydın SELÇUK

Date: APRIL 2025

With the acceleration of technological developments, cyber attacks and malware threats are becoming increasingly complex and dangerous. This situation demonstrates the necessity of developing innovative approaches and effective solutions in the field of information security. This thesis aims to enhance the effectiveness of dynamic methods in malware analysis by focusing particularly on the detailed examination of Windows operating system event log data. The primary objective of the study is to extract meaningful features from the operating system log data generated during the runtime of malicious and benign software in isolated virtual environments and to develop an innovative approach for malware detection using machine learning techniques.

To this end, executable files, determined to be either malicious or benign through both manual and online verification processes, have been meticulously collected. The data obtained from experimental applications conducted in controlled virtual machine environments were primarily classified as numerical, contextual, time-based, and statistical features. In determining which of these features could be utilized, some were further refined through feature engineering techniques, thereby creating critical indicators that enhance model performance.

The relevant features were processed using various classification algorithms such as Gradient Boosting, Logistic Regression, and Support Vector Machines; the performance of the resulting models was rigorously evaluated using the K-Fold cross-validation method. The applied data normalization and feature engineering techniques not only increased the overall accuracy of the models but also clarified the distinction between malicious and benign software. This comprehensive analysis has demonstrated that dynamic analysis-based approaches provide higher accuracy rates and lower error margins compared to traditional static methods.

Furthermore, this study aims to make a significant contribution to the field of dynamic analysis, an area that has been relatively underrepresented in the literature. Comprehensive experiments conducted on different datasets have revealed that the developed methodology can offer a proactive approach against cyber attacks. Dynamic behavior analysis yields effective results over extended time intervals in detecting both old and new malware, thereby playing a critical role in the development of cybersecurity strategies. In conclusion, this thesis illustrates the applicability and effectiveness of dynamic analysis methods in malware detection and prevention, opening new horizons in the field of cybersecurity.

Keywords: Malware, Dynamic analysis, Windows event logs, Machine learning, Virtual environment, Feature engineering, Malware detection, Cybersecurity

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Dr. Mesut GÜVEN, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim.





İÇİNDEKİLER

	<u>Sayfa</u>
ÖZET	vii
ABSTRACT	ix
TEŞEKKÜR	xi
İÇİNDEKİLER	xiii
ŞEKİL LİSTESİ	xv
ÇİZELGE LİSTESİ	xvii
ALGORİTMA LİSTESİ	xix
KAYIT LİSTESİ	xxi
KISALTMALAR	xxiii
1. GİRİŞ	1
1.1 Çalışmanın Motivasyonu ve Amacı	1
1.2 Çalışmanın Kapsamı ve Yöntemi	3
1.3 Çalışmanın Yenilikçi Yönleri	3
1.4 Tezin Genel Yapısı	3
2. İLGİLİ ÇALIŞMALAR	5
2.1 Statik ve Dinamik Analiz	5
2.2 Windows Olay Kayıtları	6
2.3 Korumalı Sanal Alan (Sandbox) Ortamları ve Kaçınma Teknikleri	6
2.4 Öznitelik Çıkarımı ve Mühendisliği	7
2.5 Veri Seti Kaynakları ve Oluşturulması	8
2.6 Makine Öğrenmesi ve Derin Öğrenmesi	9
3. METODOLOJİ VE DENEYSEL DÜZENEK	11
3.1 Veri Seti Oluşturma	11
3.1.1 Zararlı Örneklerinin Toplanması	12
3.1.2 Zararsız Yazılımların Toplanması	14
3.2 Veri Çalıştırma Süreci	16
3.2.1 Kayıt Toplama Yöntemleri	16
3.2.2 Otomatik İşlem ve Deneysel Akış	18
3.3 Sanal Ortam ve İzolasyon Teknikleri	19
3.3.1 Kaynak Tahsisi	20
3.3.2 Donanım İzi Gizleme	21
3.3.3 İzole Edilmiş Sanal Ağ	21
3.3.4 Ortam Göstergelerinin Gizlenmesi	22
3.3.5 Yazılım Göstergeleri	23
3.3.6 Ek Ağ ve Sistem Araçları	23
3.4 Deneysel Kurulum ve Çalışma Akışı	24
3.4.1 İlk İzlenim	25
3.4.2 İkinci İzlenim	31
4. ÖZNİTELİK ÇIKARIMI VE MÜHENDİSLİĞİ	35
4.1 Sayısal Bazlı Öznitelik Analizi	36
4.2 Bağlamsal Analiz	40
4.3 Örüntü Analizi	41
4.4 Anormal Analizi	41
4.5 Söz Dizimi, Entropi ve Karmaşıklık Analizi	42

4.6 Öznitelik Seçimi	42
4.6.1 Sayısal Bazlı Öznitelikler	42
4.6.2 Bağlamsal Öznitelikler	43
4.6.3 Zaman Tabanlı Öznitelikler	45
4.6.4 İstatistiksel Öznitelikler	45
4.6.5 Söz Dizimi Tabanlı Öznitelikler	46
4.6.6 Örüntü Bazlı Öznitelikler	46
4.7 Öznitelik Değerlendirme	47
4.8 Zayıf Öznitelikleri Eleme	48
4.9 Özniteliklerin Birbiriyle Olan Korelasyonlarının İncelenmesi	50
4.10 Yüksek Korelasyon İlişkisi İçeren Öznitelikleri Düzeltme	51
4.11 Tekrar Eden Özniteliklerin Kaldırılması veya Birleştirilmesi	52
4.12 Basit Modeller ile Test Etme	52
5. MODEL EĞİTİMİ VE DEĞERLENDİRME	55
5.1 K-Fold Çapraz Doğrulama	56
5.2 Kullanılan Makine Öğrenmesi Modelleri Ön Bilgi	56
5.2.1 Random Forest	56
5.2.2 Logistic Regression	57
5.2.3 Support Vector Classifier (SVC)	57
5.2.4 K-Nearest Neighbors (KNN)	57
5.2.5 Gradient Boosting	57
5.2.6 RobustScaler	58
5.2.7 MinMaxScaler	58
5.2.8 Sinir Ağları	58
5.2.9 MLP Sınıflandırıcısı	59
5.2.10 Değerlendirme Metrikleri	59
5.2.11 Aykırı Değerlerin (Outlier) Temizlenmesi	61
5.3 Sonuçların Yorumlanması	62
5.3.1 Aykırı Değer Temizleme Değerlendirme	63
5.3.2 Model Sonuçları	63
5.3.3 Veri Küme Sonuçları	65
5.3.4 En İyi Model Çıkarımları	66
6. SONUÇLAR VE TARTIŞMA	69
6.1 Sandbox Kurulumu ve Kaçınma Teknikleri	69
6.2 Veri Toplama ve Otomasyon Zorlukları	69
6.3 Öznitelik Çıkarım Süreci	70
6.4 Model Eğitimi ve Değerlendirme	70
6.5 Derin Öğrenme ve Doğal Dil İşleme Gelecek Çalışmalar	71
6.6 Sonuç ve Tartışma	72
KAYNAKLAR	73

ŞEKİL LİSTESİ

	Sayfa
Şekil 3.1: Veri Seti Diyagramı.	16
Şekil 3.2: Deneysel Akış.	20
Şekil 3.3: Tamamlanmış Deneysel Akış.	31
Şekil 3.4: Sürelere Göre Kayıt Boyutları.	32
Şekil 4.1: Öznitelikler Korelasyon Matris Değişimi.	53





ÇİZELGE LİSTESİ

	Sayfa
Çizelge 3.1: Zararlı Veri Seti.	14
Çizelge 3.2: Zararsız Veri Seti.	15
Çizelge 3.3: Çalıştırılan Bir Dosyanın Kayıtları.	18
Çizelge 3.4: İlk İzlenimde Zararsız ve Zararlı Dosyaların Oluşturduğu Sorunlar.	27
Çizelge 4.1: Kayıt Verilerinden Çıkarılan Özniteliklerin Dosya ve Kayıt Sayıları.	37
Çizelge 4.2: Kayıt Verilerinden Çıkarılan En Yüksek Eşsiz Anahtar Kelime Zararlı Dosya Oranları.	38
Çizelge 4.3: Zararlı ve Zararsız Yazılımlarda Bazı Anahtar Kelimelerin Benzer-siz Değer Sayısı.	39
Çizelge 4.4: Zararlı Yazılımlar Tarafından Kullanılan Şüpheli Komut Argümanları.	44
Çizelge 4.5: Öznitelik Korelasyon Değerleri.	48
Çizelge 4.6: Yüksek Korelasyona Sahip Öznitelik Çiftleri.	51
Çizelge 5.1: Veri kümelerine ait özet istatistikler.	56
Çizelge 5.2: Aykırı değer temizleme kurallarının her küme üzerindeki etkisi.	64
Çizelge 5.3: Sınıflandırıcıların ortalama (<i>ort</i>) ve standart sapma (<i>ss</i>) değerleri – aşırı öğrenme artan sırada.	64
Çizelge 5.4: Veri kümelerinin ortalama (<i>ort</i>) ve standart sapma (<i>ss</i>) değerleri – aşırı öğrenme artan sırada.	66
Çizelge 5.5: En İyi Model performans metrikleri — <i>En Düşük Başarı</i> azalan sırada.	67
Çizelge 5.6: Seçili konfigürasyonların ortalama karışıklık matrisi bileşenleri ve başlıca performans metrikleri.	67



ALGORİTMA LİSTESİ

	Sayfa
Algoritma 1 : Bir Çalıştırılabilir Dosyanın Kayıtlarının Toplanma Süreci. . .	17
Algoritma 2 : Örnek BAT Dosyası Kodu.	30
Algoritma 3 : VM Anlık Görüntü Yönetimi ile Dosya Çalıştırma Süreci. . . .	30





KAYIT LİSTESİ

	<u>Sayfa</u>
Kayıt 1 : Örnek JSON Kaydı.	36





KISALTMALAR

ML	: Makine Öğrenmesi (Machine Learning)
DL	: Derin Öğrenme (Deep Learning)
VM	: Sanal Makine (Virtual Machine)
CNN	: Konvolüsyonel sinir ağları (Convolutional Neural Network)
Bi-LSTM	: Çift Yönlü Uzun Kısa Süreli Bellek (Bidirectional Long Short Term Memory)
API	: Uygulama Programlama Arabirimi (Application Programming Interface)
BERT	: Çift Yönlü Kodlayıcı Temsilciler (Bidirectional Encoder Representations from Transformers)
RNN	: Yinelenen Sinir Ağı (Recurrent Neural Network)
SVC	: Destek Vektör Makinesi (Support Vector Machine)
HDD	: Sabit Disk Sürücü (Hard Disk Drive)
SSD	: Katı Hal Sürücü (Solid State Disk)
RAM	: Rastgele Erişimli Bellek (Dynamic Random Access Memory)
CPU	: Merkezî işlem birimi (Central Processing Unit)
JSON	: Javascript Nesne Notasyonu (JavaScript Object Notation)
XML	: Genişletilebilir İşaretleme Dili (Extensible Markup Language)
HTML	: Hiper Metin İşaretleme Dili (HyperText Markup Language)
HTTP	: Hiper Metin Transfer Protokolü (HyperText Transfer Protocol)
FTP	: Dosya Transfer Protokolü (File Transfer Protocol)
SMTP	: Basit Posta Transfer Protokolü (Simple Mail Transfer Protocol)
BIOS	: Temel Giriş/Çıkış Sistemi (Basic Input/Output System)
DNS	: Alan Adı Sistemi (Domain Name System)
XGBoost	: eXtreme Gradient Boosting
KNN	: K En Yakın Komşu (K-Nearest Neighbors)
GNN	: Grafik sinir ağları (Graph neural networks)
RFE	: Özyinelemeli öznelilik Eleme (Recursive Feature Elimination)
PE	: Taşınabilir Yürütülebilir Dosya (Portable Executable)
MLP	: Çok Katmanlı Algılayıcı (Multi-Layer Perceptron)
ReLU	: ReLU (Rectified Linear Unit)
ROC AUC	: Alıcı İşletim Karakteristiği Eğrisi Altındaki Alan (Receiver Operating Characteristic - Area Under Curve)



1. GİRİŞ

1.1 Çalışmanın Motivasyonu ve Amacı

Günümüzde hızla artan siber saldırılar, kurumsal ve bireysel sistemlerin güvenliğini tehdit eden en kritik unsurlardan biri haline gelmiştir. Zararlı yazılımlar, bu saldırıların temel araçları arasında yer almakta ve çeşitli saldırı tipleri ile davranışlar sergilemektedir. İlk dönemlerde nispeten basit yapılarla geliştirilen virüsler, trojanlar gibi zararlı yazılım türleri, günümüzde çok katmanlı, çeşitli gizlenme ve kaçınma teknikleri kullanan karmaşık yazılımlara dönüşmüştür. Kötü niyetli yazılım, bazı türleri sistemlere doğrudan zarar verirken, bazıları bulaşıcı olarak diğer cihazlara yayılmaktadır. Virüsler, solucanlar, trojanlar ve fidye yazılımları gibi türler, ev sahibi cihazlara karşı yüksek düzeyde saldırganlık gösterirken, adware fiziksel zarar vermeden geliştiricilere reklam gelirleri sağlar [1]. Keylogger'lar ise, cihazdaki kişisel ve kurumsal bilgileri toplayarak, cihazdan ziyade cihazın sahibine yönelik tehdit oluşturmaktadır. Dolayısıyla, yazılım türleri davranışsal olarak büyük farklılıklar sergilemekte ve tespit süreçlerinde ortak örüntülerin belirlenmesi hayati önem taşımaktadır.

Geleneksel statik analiz yöntemleri, yazılımın kodunu çalıştırmadan inceler. Modern yazılımlar, ancak çalışma anında ortaya çıkan saklı davranışları ile tespit edilebilmektedir. Bu sebeple, zararlı yazılımların yalnızca kod düzeyinde değil, çalışma anındaki dosya, ağ ve sistem etkileşimlerini gözler önüne seren dinamik analiz yöntemleri ön plana çıkmaktadır. Bununla birlikte, pek çok zararlı yazılım türü, analiz ortamlarını tespit edebilecek kadar gelişmiş kaçınma stratejileri sergilemektedir. Bu durum, dinamik analiz süreçlerinin de sürekli yenilikçi yaklaşımlarla desteklenmesini ve uzun vadeli etkili yöntemlerin geliştirilmesini zorunlu kılmaktadır. Dinamik analizin, zararlı yazılımların davranışlarını örüntü bazında yönetebilmesi, ileri aşamada ortaya çıkabilecek yeni zararlı yazılımların tespitinde önemli rol oynayabilir.

Özellikle Windows işletim sistemine ait olay kayıt verileri, kullanıcı giriş-çıkışları, uygulamaların başlatılması veya sonlandırılması gibi pek çok sistemsel olayı ayrıntılı

biçimde barındırması sebebiyle, dinamik analizde zengin bir veri kaynağı olarak değerlendirilmektedir. Ancak bu verilerin yüksek hacmi ve çeşitliliği, geleneksel yöntemlerle etkin analiz edilmesini güçleştirmekte, bu nedenle otomasyon ve örüntü tanımaya odaklı makine öğrenmesi tekniklerinin kullanımını zorunlu hale getirmektedir.

Zararlı ve zararsız yazılımların davranışlarını daha net ortaya koymak, siber saldırılara karşı proaktif bir savunma mekanizması geliştirmek ve literatürde henüz yeterince incelenmemiş olan Windows olay kayıt verilerinin potansiyelini ortaya çıkarmak, çalışmanın temel motivasyon kaynaklarıdır.

Bu çalışmanın temel amacı, Windows işletim sistemine ait olay kayıt verilerinden elde edilen dinamik veriler aracılığıyla, yazılım tespitinde mevcut yöntemlerin yetersizliklerini aşan, yüksek doğruluk oranlarına sahip yenilikçi bir sistem geliştirmektir. Çalışmada, izole edilmiş sanal ortamlarda zararlı ve zararsız yazılımların oluşturduğu kayıt verileri üzerinden, makine öğrenmesi tekniklerinin uygulanması hedeflenmektedir. Böylece, yazılımların çalışma anındaki dosya, ağ ve sistem etkileşimlerinin detaylı incelenmesi mümkün olacak, bu sayede zararlı davranışların ortaya çıkarılmasında statik analiz yöntemlerinin ötesinde bir yaklaşım sunulacaktır.

Bu amaç doğrultusunda, çalışma kapsamına giren aşamalardan ilki, ham kayıt verilerinin kapsamlı bir şekilde toplanması ve ön işleme tabi tutulmasıdır. Toplanan verilerden, sayısal, bağlamsal, zaman tabanlı ve istatistiksel öznitelikler çıkarılacak, bu özniteliklerin genel örüntüleri belirlenerek zararlı ile zararsız yazılımlar arasındaki farklar ortaya konulacaktır. Geliştirilecek model, Gradient Boosting, Logistic Regression, Destek Vektör Makineleri (SVC) gibi çeşitli makine öğrenmesi algoritmaları kullanılarak eğitilecek ve K-Fold çapraz doğrulama gibi yöntemlerle titizlikle değerlendirilecektir.

Çalışmanın amacı aynı zamanda, dinamik analiz yöntemlerinin zararlı yazılım tespitindeki etkinliğini artırarak, günümüzün sürekli değişen ve kendini adapte eden siber tehditlerine karşı aktif bir savunma mekanizması oluşturmak üzerine odaklanmaktadır. Geliştirilecek sistem, yalnızca zararlı ve zararsız yazılımların ayırt edilmesini sağlamakla kalmayıp, aynı zamanda farklı yazılım türlerinin davranışsal özelliklerini modelleyerek yeni tehditlerin tespitinde esnek ve adaptif bir yapı sunmayı hedeflemektedir.

Bu çalışma, Windows olay kayıt verilerinin zenginliğinden yararlanarak, makine öğrenmesi ve dinamik analiz yöntemlerini birleştiren yenilikçi bir yaklaşımla yazılım tipi tespitinde yeni bir perspektif sunmayı amaçlamakta ve böylece hem akademik literatüre hem de uygulamalı siber güvenlik çözümlerine önemli katkılar sağlamayı hedeflemektedir.

1.2 Çalışmanın Kapsamı ve Yöntemi

Bu çalışma, izole edilmiş sanal ortamlarda gerçekleştirilen deneysel uygulamalar temelinde yürütülmüştür. Araştırma kapsamında, hem manuel hem de çevrimiçi doğrulama süreçleriyle belirlenen çalıştırılabilir dosyalardan Windows olay kayıt verileri toplanmıştır. Bu veriler sayısal, bağlamsal, zaman tabanlı ve istatistiksel öznitelikler olarak sınıflandırılmıştır. Çalışmada, veri ön işleme, öznitelik çıkarımı, model eğitimi ve değerlendirme aşamaları titizlikle uygulanmıştır. Gradient Boosting, Logistic Regression, Destek Vektör Makineleri (SVM) ve diğer sınıflandırma algoritmaları kullanılarak, elde edilen veriler üzerinden yazılım tipi tespitine yönelik modeller oluşturulmuş ve K-Fold çapraz doğrulama yöntemi ile performansları ölçülmüştür.

1.3 Çalışmanın Yenilikçi Yönleri

Çalışmanın yenilikçi yönleri, dinamik analiz yöntemlerinin Windows olay kayıt verileriyle entegre edilmesi ve makine öğrenmesi tekniklerinin bu veriler üzerinden uygulanmasıdır. Geleneksel statik analiz yöntemlerinin ötesine geçilerek, zararlı yazılımların çalışma anındaki davranışlarının detaylı bir şekilde incelenmesi sağlanmaktadır. Bu yaklaşım, analiz ortamlarını tespit edebilen gelişmiş tehlikeli program türlerine karşı daha adaptif ve uzun vadeli etkili bir savunma mekanizması sunmaktadır. Ayrıca, literatürde henüz yeterince ele alınmamış olan veri setlerinin potansiyelini ortaya koymaktadır.

1.4 Tezin Genel Yapısı

Tezin ilerleyen bölümlerinde, şu ana başlıklar detaylı olarak ele alınacaktır:

- **İlgili Çalışmalar:** Zararlı yazılım analizi yöntemleri, statik ve dinamik analiz teknikleri, makine öğrenmesi ve derin öğrenme uygulamaları ile mevcut yakla-

şımların eleştirileri.

- **Metodoloji ve Deneysel Düzenek:** Veri seti oluşturma (zararsız yazılımların ve zararlı yazılım örneklerinin toplanması), veri toplama süreci (kayıt toplama yöntemleri, otomatik işlem ve deneysel akış), sanal ortam ve izolasyon teknikleri, deneysel kurulum ve çalışma akışı.
- **Öznitelik Çıkarımı ve Mühendislik:** Kayıt verilerinden öznitelik çıkarım yöntemleri, sayısal, bağlamsal ve zaman tabanlı öznitelikler, istatistiksel ve söz dizimi (syntax) tabanlı öznitelikler, öznitelik seçimi ve model optimizasyonu.
- **Model Eğitimi ve Değerlendirme:** Makine öğrenmesi modelleri (Random Forest, Logistic Regression, SVM, K-Nearest Neighbors (KNN), Gradient Boosting), derin öğrenme yaklaşımları, K-Fold çapraz doğrulama ve performans metrikleri, sonuçların karşılaştırılması ve yorumlanması.
- **Sonuçlar:** Çalışmanın genel değerlendirmesi, sınırlamalar ve tartışmalar.

2. İLGİLİ ÇALIŞMALAR

2.1 Statik ve Dinamik Analiz

Yazılım tipi tespiti, zararlı yazılımların hızla gelişen doğasıyla başa çıkabilmek amacıyla sürekli araştırılan bir alan olup, bu alanda yenilikçi çözümler ortaya konulmaktadır. Yıllar içinde, araştırmacılar sezgisel ve kural tabanlı sistemlerden makine öğrenmesi ve derin öğrenme yaklaşımlarına kadar çeşitli yöntemler önermiştir. Her biri kendine özgü avantajlar sunar.

Geleneksel zararlı yazılım analiz teknikleri genel olarak **statik analiz** ve **dinamik analiz** olarak iki ana kategoriye ayrılır. Statik analiz, bir zararlı yazılım örneğinin kodunu, dosya yapısını ve meta verilerini çalıştırılmadan inceleyerek bilinen kötü niyetli imzalar [2], hash tabanlı karşılaştırmalar ve sezgisel kuralları hızlıca tespit edebilme avantajına sahiptir. Ancak, şifreleme gibi karmaşıklıklaştırmaya (obfuscation) teknikleri, kötü niyetli yükü gizleyerek statik analizin etkinliğini düşürmektedir [2].

Öte yandan, **dinamik analiz**, tehlikeli programların kontrollü bir ortamda sanal makineler çalıştırılarak çalışma zamanı davranışlarının izlenmesini içerir. Bu yöntem, API çağrıları, ağ iletişimi ve kayıt defteri değişiklikleri gibi dinamik özellikleri yakalayıp kötü niyetli davranışlar hakkında derinlemesine bilgiler sunar [3]. Örneğin, **DMalNet** çerçevesi, dinamik API çağrıları ve özellik mühendisliğini kullanarak yüksek doğrulukla yazılım tipi sınıflandırması yapmaktadır [4]. Ayrıca, dinamik tekniklerin, sistem çağrıları, bellek tahsisi ve süreç oluşturma desenlerini tespit etmede etkili olduğu da kanıtlanmıştır [5]. Ancak dinamik analiz, yüksek hesaplama maliyeti ve analiz ortamını tespit etme riskleri nedeniyle bazı zorluklar da barındırmaktadır.

Bu sınırlamaları aşmak amacıyla, statik ve dinamik özellikleri birleştiren **hibrit yaklaşımlar** geliştirilmiştir. Hibrit yöntemler, her iki tekniğin güçlü yönlerini harmanlayarak daha doğru ve sağlam yazılım türü tespiti sağlar. Örneğin, bazı hibrit çerçeveler, statik analizden elde edilen opcode dizilerini dinamik analizden elde edilen API çağrı izleri

ile birleřtirerek tamamlayıcı bilgileri kullanır. Ayrıca, özellik mühendislięi ve boyut indirgeme tekniklerindeki gelişmeler, birleşik veri setlerindeki gürültüyü azaltarak hibrit yaklaşımların performansını artırmıştır [6].

Son zamanlarda, makine öğrenmesi ve derin öğrenme yaklaşımlarının entegrasyonu zararlı yazılım tespitinde önemli bir sıçrama yaratmıştır. Sinir ağı tabanlı sınıflandırıcılar [7], pekiştirmeli öğrenme [8] ve grafik tabanlı analiz [9] gibi teknikler, hem tespit doğruluğunu hem de yeni tehlikeli program ve yazılım varyantlarına uyum sağlama kabiliyetini artırmada etkili olmuştur. Bununla birlikte, yanlış pozitifleri azaltma, ölçeklenebilirlik ve saldırganların benimseyebileceęi kaçınma teknikleri gibi zorluklar halen devam etmektedir. Statik ve dinamik özelliklerin hibritleştirilmesi, zararlı yazılım davranışını daha kapsamlı anlamada güçlü bir çözüm sunmakta ve bu yaklaşım giderek daha fazla tercih edilmektedir [10].

2.2 Windows Olay Kayıtları

Steverson ve arkadaşlarının çalışmasında [11], işletim sistemi tarafından tutulan "Windows Event Log" girdileri doğal dil işleme (NLP) teknikleriyle inceleniyor. Yani her bir kayıt, bir metin parçası gibi ele alınıp makineye "hangi kelimeyi hangi kelime izler?" sorusu yöneltiliyor. Yalnızca üzerinde çalıştığı bilgisayardan gelen kayıtlara ihtiyaç duyan bu yöntem, yüksek doğruluk ve düşük yanlış alarm oranıyla, gerçek zamanlı "uç nokta" (endpoint) savunma sistemlerine doğrudan entegre edilebilecek pratik bir çözüm sunuyor. Fakat çok yüksek başarı oranları modelin genelleme yapabilme yeteneęi ile alakalı şüpheye sebep oluyor.

2.3 Korumalı Sanal Alan (Sandbox) Ortamları ve Kaçınma Teknikleri

Korumalı Sanal Alan (sandbox), zararlı yazılım analizinde kullanılan, gerçek sistemden izole edilmiş kontrollü bir sanal ortamdır. Bu ortam, potansiyel zararlı yazılımların çalıştırılmasına izin vererek, onların davranışlarını güvenli bir şekilde gözlemlemeye olanak tanır. Sandbox¹ sayesinde, yazılımın tehlikeli davranışlarının sistemde kalıcı hasar vermeden, izole edilmiş bir ortamda çalışması sağlanır ve elde edilen kayıt verileri, davranışlarının detaylı analizinde kullanılır [12]

¹Sandbox: Korumalı Sanal Alan

Zararlı yazılımlar, dinamik analiz sırasında sandbox ortamını tespit etmek ve analizden kaçınmak için çeşitli ileri teknikler kullanmaktadır. Bu saklanma (evasion) teknikleri, saldırganların kötü niyetli faaliyetlerini gizlemek için kritik öneme sahiptir. Sharma ve arkadaşları (2022), kod enjeksiyonu, izleme süresini aşmak için zaman gecikmeleri ve sanallaştırılmış ortamı tespit etmek amacıyla çevresel parmak izi alma gibi saklanma tekniklerini sınıflandırmıştır [13]. Saldırganlar, belirli donanım veya işletim sistemi öğelerini sorgulayarak sandbox ortamını tespit etmeye çalışırken, bu bulgular, sandbox'ların varlıklarını ifşa eden izleri minimize edecek şekilde tasarlanması gerekliliğini vurgulamaktadır.

Lusky ve Mendelson (2021) tarafından yapılan bir çalışmada, donanım yan kanalları kullanılarak gerçekleştirilen zamanlama saldırıları incelenmiş ve belirli talimatların çalışma süreleri ölçülmüştür [14]. CPU çekirdek planlamasına dayalı önerilen çözümler, zaman bilgisini gizleyerek agresif programların bu bilgiye dayanarak analiz ortamını tespit etmesini zorlaştırmaktadır. Birçok mevcut sandbox aracı, örneğin Cuckoo [15], statik konfigürasyonlara dayalı çalıştıkları için, zararlı yazılımın tutarlı çevresel göstergeler arayarak ortamı tespit etme riski taşımaktadır [16]. Bu nedenle, MORRIGU gibi yeniden yapılandırılabilir ortamlar geliştirilmiştir. MORRIGU, zararlı yazılımın kaçınma veya saklanma stratejilerine meydan okuyarak dinamik olarak uyarlanabilen bir yapı sunar. Örneğin, rastgele zaman atlamaları, sistem araçlarının devre dışı bırakılması ve işletim ortamının değiştirilmesi gibi tekniklerle analiz ortamının tespit edilmesini zorlaştırmaktadır. Tehlikeli program geliştiricileri kaçınma tekniklerini sürekli olarak geliştirdikçe, sandbox ortamlarının gerçek zamanlı uyarlanabilirlik, donanım tabanlı savunma ve yapay zeka destekli davranışsal analiz mekanizmaları ile güçlendirilmesi gerekmektedir.

2.4 Öznitelik Çıkarımı ve Mühendisliği

Doğru öznitelik çıkarımı, zararlı yazılım tespitinde temel rol oynamakta ve kötü niyetli davranışların belirlenmesinde kritik girdileri sağlamaktadır. Android zararlı yazılımlarının tespitinde, izinler, API çağrı dizileri ve uygulama bazlı anahtar (intent) filtreleri gibi öznitelikler, bir uygulamanın hassas verilere erişimi veya yetkisiz işlemler gerçekleştirmesi gibi önemli davranışları ortaya koymaktadır [17]. Dinamik izlerin statik analizle birleştirilmesi, hibrit öznitelik setleri oluşturma yoluyla tespit oranlarını artır-

mada etkili bir yöntemdir.

Web ve masaüstü platformlarında, Cuckoo Sandbox gibi araçlar, tehlikeli programların çalışma zamanı davranışlarını ayrıntılı olarak kaydetmekte, dosya işlemleri, kayıt defteri değişiklikleri ve ağ iletişimi gibi sistem etkileşimlerini gözlemlemektedir [18]. Ağ etkileşimlerinin JSON veya XML formatında dışa aktarılması, sonrasında verilerin etkin şekilde işlenmesine olanak tanımakta ve anormal veri akışları, alışılmadık DNS sorguları veya şüpheli HTTP istekleri gibi desenlerin tespit edilmesini kolaylaştırmaktadır.

Ham verinin anlamlı temsillere dönüştürülmesinde, veri normalizasyonu, boyut indirgeme ve "embedding" teknikleri kullanılmaktadır. Örneğin, DMalNet yaklaşımı, API çağrılarını veya ağ etkinliklerini Word2Vec kullanarak sayısal vektörlere dönüştürmekte ve böylece verideki semantik benzerlikleri yakalamaktadır [19]. Ayrıca, grafik sinir ağları (GNN) kullanılarak, sistem süreçleri veya ağ paketleri arasındaki ilişkiler modellenmekte, bu da daha derin bağlamsal desenlerin ortaya çıkarılmasına olanak tanımaktadır.

Öznitelik çıkarımı aşamasında, gereksiz özniteliklerin çıkarılması, modelin performansını ve genelleme yeteneğini artırmak için kritik öneme sahiptir. Özyinelemeli Öznitelik Eleme (RFE) ve karşılıklı bilgi seçimi gibi yöntemler, en bilgilendirici özniteliklerin seçilmesine yardımcı olmaktadır. Ayrıca, AutoML araçları gibi gelişmiş yöntemler, karmaşık öznitelik çıkarım görevlerinde ölçeklenebilir çözümler sunarak zararlı yazılım tespitinin doğruluğunu artırmaktadır.

Gelecekte, alan bilgisi ile ileri yapay zeka tekniklerini entegre eden hibrit yaklaşımların, verideki gizli ilişkileri daha iyi ortaya çıkararak tespit doğruluğunu artıracakları öngörülmektedir.

2.5 Veri Seti Kaynakları ve Oluşturulması

Yüksek kaliteli ve çeşitlilik gösteren veri setlerinin oluşturulması, zararlı yazılım tespitinde etkili araştırmaların temel taşlarından biridir. İyi düzenlenmiş bir veri seti, tespit modellerinin güvenilirliğini, temsil edilebilirliğini ve genelleme yeteneğini artırır. Zararlı ve Zararsız yazılım örneklerin toplanmasında, hem halka açık depo kaynakları

hem de özel veri setleri kullanılmıştır.

Örneğin, DMalNet çalışması, en yaygın zararlı yazılım formatı olan Windows taşınabilir yürütülebilir dosya (PE) dosyalarını toplamak için VirusShare [20] üzerinden otomatik günlük indirme betikleri kullanmış ve veri setini sürekli güncel tutmuştur. Güvenli örnekler için ise, PortableApps [21], Softonic [22], SourceForge [23] ve CNET [24] gibi saygın yazılım depoları tercih edilmiştir. Her bir örneğin VirusTotal [25] ile çapraz doğrulaması yapılarak, dosyaların zararlı ya da güvenli olduğu tespit edilmiştir [4]. Bu doğrulama adımı, veri setinde yanlış etiketlenmiş örneklerin yer almamasını sağlamış. Eski zararlı yazılım örnekleri ile yeni tehditlerin harmanlanması, gerçek dünya senaryolarını yansıtacak hibrit bir veri seti oluşturulmasına olanak tanımıştır. Diğer çalışmalar, zararsız Android uygulamaları için APKMirror ve zararlı yazılım veri setleri için Androzoo gibi kaynaklardan faydalanmışlardır [26].

Veri seti doğrulaması, VirusTotal gibi platformlar aracılığıyla gerçekleştirilen kapsamlı örnek doğrulamaları ile desteklenmekte ve güncel tehditleri yansıtacak gerçek zamanlı veri toplama yaklaşımları giderek önem kazanmaktadır [27, 28].

2.6 Makine Öğrenmesi ve Derin Öğrenmesi

Makine öğrenmesi (ML) ve derin öğrenmesi (DL) yaklaşımları, son yıllarda zararlı yazılım tespitinde önemli ilerlemeler kaydetmiştir. Bu modeller, karmaşık davranışsal desenleri tanıma, yüksek boyutlu verilerle başa çıkmada ve dinamik analiz verilerinden anlamlı bilgileri çıkarmada son derece etkilidir. Çalışmalar, bu yöntemlerin veri setine ve öznitelik temsiline bağlı olarak %90 ile %98 arasında doğruluk oranlarına ulaşabildiğini göstermiştir [4]. Örneğin, Convolutional Neural Network (CNN), yerel desenleri tanıma etkili olurken, Bi-LSTM gibi tekrarlayan mimariler, zararlı yazılım davranışlarındaki zamansal bağımlılıkları yakalamada kullanılmaktadır [29]. Hibrit modeller, zamansal ve mekânsal öznitelik çıkarımını birleştirerek, tek başına kullanılan mimarilere göre üstün performans göstermektedir. Örneğin, CNN'lerin grafik sinir ağları ile entegrasyonu, süreçler arası ilişkilerin modellenmesinde etkileyici sonuçlar sunmaktadır [30].

Derin sinir ağları, yüksek boyutlu verilerde gerekli görülmeyen bilgileri tolere edebilmekte ve karmaşık ilişkileri öğrenebilmektedir. Buna ek olarak, gradyan arttırmalı

karar ağaçları, tablosal veri üzerinde verimlilik ve dayanıklılık açısından yaygın olarak tercih edilmektedir. Elnaggar ve arkadaşları (2022)'nin çalışması, statik ve dinamik özniteliklerin XGBoost ile birleştirilmesinin, derin öğrenme modellerinin getirdiği karmaşıklığa ihtiyaç duymadan yüksek performans elde edilebildiğini göstermiştir [31].

Windows olay kayıtları, yazılım tespitinde zengin davranışsal veriler sunar. Dinamik olarak çıkarılan öznitelikler, anormal davranışların tespitinde kritik rol oynamaktadır [32]. Son zamanlarda, "Transformer" tabanlı modeller, örneğin BERT varyantları, sistem çağrıları ve dosya etkileşimleri arasındaki uzun menzilli bağımlılıkları yakalamada geleneksel RNN'lerden üstün performans sergilemiştir [33]. Ayrıca, karşıt öğrenme yöntemleri, etiketli veri gereksinimini azaltarak düşük veri senaryolarında bile etkili tespit sağlamaktadır [34].

Ancak, gerçek dünya uygulamalarında yanlış pozitif oranlarını azaltmak ve ölçeklenebilirliği sağlamak halen önemli zorluklardandır. Geleneksel ML yöntemleri ile DL yaklaşımlarının hibritleştirilmesi, örneğin, gradyan arttırmalı ağaçlar ile GNN'lerin entegrasyonu, farklı veri setleri üzerinde modelin genelleme yeteneğini artırmada etkili sonuçlar sunmaktadır [30]. Ayrıca toplu öğrenme teknikleri, birden fazla modelin tahminlerinin birleştirilmesiyle tespit performansını daha da iyileştirmektedir [35].

Genel olarak, makine öğrenmesi ve derin öğrenmesi yöntemlerinin gelişmekte olan yapısı, yazılım tespitinde karşılaşılan karmaşık zorlukların üstesinden gelebilmek için büyük bir potansiyele sahiptir. Dinamik olarak çıkarılan öznitelikler, hibrit mimariler ve dikkat mekanizmaları sayesinde, tespit doğruluğu ve ölçeklenebilirlik sürekli olarak artırılmakta ve gelişmiş tehlikeli programların tehditleriyle mücadelede önemli adımlar atılmaktadır.

3. METODOLOJİ VE DENEYSEL DÜZENEK

Bu bölümde, dinamik zararlı yazılım davranış analizinin gerçekleştirilmesi için kullanılan yöntemler, deneysel düzenek ve veri toplama süreci detaylandırılmıştır. Bu tezde amaçlanan, Windows olay kayıtlarından elde edilen verilerle zararlı yazılım tespitini daha doğruluklu hale getirecek bir model geliştirmektir. Bunun için öncelikle ihtiyaç olan süreçlerin analiz edilmesi gerekmektedir.

Süreçlerde, ilk adım olarak, Windows olay kayıt verilerinin ve ilgili veri kaynaklarının standart bir formata getirilmesi gerekmektedir. Bu aşamada, kayıtların hangi anahtar metrikler üzerinden inceleneceği, hangi parametrelerin çıkarılacağı ve bu verilerin nasıl yapılandırılacağı belirlenmelidir.

Bunun ardından, elde edilen verilerin otomatik olarak toplanması, kaydedilmesi ve yönetilmesi süreci devreye girecektir. Verilerin sürekli izlenmesi ve merkezi bir veri deposunda birleştirilmesi, analiz sonuçlarının güncelliğini ve güvenilirliğini sağlayacaktır.

Bu temel süreçlerin tamamı, daha ayrıntılı olarak alt bölümlerde ele alınacaktır. Ancak genel hatlarıyla, analiz için çalışılması gereken unsurların öncelikle belirli bir formata oturtulması, sonrasında ise otomatik yönetim mekanizmaları ile düzenli olarak toplanması kritik adımlardır.

3.1 Veri Seti Oluşturma

Veri seti oluşturma sürecinde, çalışmanın temel gereksinimlerine odaklanmak büyük önem taşımaktadır. Bu çalışmanın ana hedefi, yazılım tiplerini doğru biçimde ayırt edebilmek, yani yazılımın zararlı mı yoksa zararsız mı olduğunu belirleyebilmektir. Bu hedefe ulaşabilmek için, her yazılım tipine ait ayrıntılı bilgi toplanması gerekmektedir. Literatürde bazı çalışmalar, zararsız yazılımları alt kategorilere ayırarak incelemeyi hedeflese de [4], burada sadece iki sınıf olan zararlı ve zararsız üzerinde yoğunlaşmıştır. Bu tercih, Windows olay kayıtlarının yeterli bilgi sunmaması değil, öncelikli

olarak yazılımın zararsız olup olmadığını doğru tespit etmenin temel hedeflenmesi ve bundan sonraki çalışmalar için sağlam bir temel oluşturulması amacıyla yapılmıştır.

Ayrıca, zararlı ve zararsız yazılımların tespiti sırasında, hangi dosya formatında veri toplanacağını ve çalışmanın hangi domaine hitap edeceğini belirlenmesi önemlidir. Çalışma Windows olay kayıtlarına dayalı olduğundan, hedef kitle şimdilik yalnızca Windows işletim sistemine yönelik olacaktır. Bu bağlamda, çalıştırılabilir dosya tipleri veri setimizin temelini oluşturur. Windows'ta programların en yaygın dağıtım biçimi olan ".exe" dosya tipleri bu yüzden kritik öneme sahiptir. Ancak, sadece exe dosyalarına odaklanmak, programın tüm davranışlarının tam olarak gözlemlenmesini engelleyebilir. Çünkü bir program, exe dosyasının yanı sıra, yanındaki diğer dosya tiplerini de içerebilir ve exe aracılığıyla bu dosyalar da çalıştırılmaktadır. Dolayısıyla, exe dosyasıyla birlikte, programa ait tüm dosyaların aynı dizin altında bulunması, analiz için oldukça önemlidir.

Son olarak, çalıştırılacak dosya sayısında belirli bir eşik değeri hedeflemek, her iki sınıf için de yeterli ve dengeli veri toplamak açısından önemlidir. Bu çalışmada, zararlı ve zararsız yazılımlar için dosya sayısı eşik değeri olarak 2500 örnek belirlenmiştir. Sınıflar arasında görülebilecek farklılıklar, modelleme aşamasında detaylı olarak incelenecek ve sınıflandırma performansı üzerinde etkileri değerlendirilecektir.

3.1.1 Zararlı Örneklerinin Toplanması

Dinamik davranışsal analizde, çalışmanın başarısını ve elde edilecek sonuçların güvenilirliğini doğrudan etkileyen en kritik adımlardan biri, zararlı yazılım örneklerinin geniş, güncel ve doğrulanabilir bir veri seti oluşturularak temin edilmesidir. Bu çalışma kapsamında, zararlı örnekler tamamen **Malware Bazaar** [36] üzerinden temin edilmiştir.

Malware Bazaar, Abuse.ch tarafından yönetilen ve siber güvenlik araştırmacıları ile uzmanlarına kapsamlı zararlı yazılım örnekleri sunan açık kaynaklı bir veri deposudur. Platform, zararlı yazılım örneklerini sadece dosya olarak sağlamakla kalmayıp, her örnek için SHA-256 gibi kriptografik özet değerleri, dosya boyutu, oluşturulma tarihi gibi detaylı meta verileri de paylaşmaktadır. Bu bilgiler, örneklerin tanımlanması, ben-

zersizliđinin sađlanması ve diđer veri tabanlarıyla apraz kontrol iřlemleri iin byk nem tařıtmaktadır.

Malware Bazaar, rnekleri saatlik olarak gncellenen bir yapıya sahiptir. Bu gncelleme dngs sayesinde, her hafta yeni ve gncel zararlı yazılım dosyaları zip formunda indirilebilir hale gelmektedir. Srekli gncellenen bu yapı, veri setinin gerek dnya senaryolarını yansıtmasını sađlar. Bylece, analiz ve modelleme srecinde kullanılan rnekler en yeni tehditleri ierir. rneđin, SHA-256 hash deđerleri, her zararlı yazılım rneđinin benzersiz olarak tanımlanmasını ve gerektiđinde tekrar eden rneklerin veya hatalı etiketlemenin kolayca tespit edilmesini mmkn kılar.

Bu yaklařım, veri seti oluřturma srecinde gncellik, eřitlilik ve dođrulanabilirlik aısından byk avantaj sađlamaktadır. Bu tr alıřmalarda eski olabilecek verisetlerinin kullanımı oluřan sonucunda eskiye ynelik olmasına neden olur. Eski zararlı bir dosyanın davranıřları ile yeni tip bir zararlı yazılım davranıřlarında farklılıklar fazla olabilir. O yzden gncel tutulabilecek bir set kullanımı alıřmanın verimini ve geerliliđini koruyacaktır.

İndirilen rneklerin gerekten zararlı olup olmadıđını varsaymak yanlış sonuca ynlendirebilir. Bu yzden Malware Bazaar'dan indirilen dosyaların statsn dođrulayacak bir mekanizmaya ihtiya bulunmaktadır. Bu dođrulama **VirusTotal** [25] ile sađlanabilir.

VirusTotal, ok sayıda antivirus motorunun sonularını bir araya getiren kapsamlı bir tarama servisi sunar. Her yazılım rneđi iin, VirusTotal raporları, hangi antivirus motorlarının zararlı olduđunu tespit ettiđini ve hangi motorların negatif sonu verdiđini gsterir. Bu sayede, Malware Bazaar'dan alınan rneklerin dođruluđu, VirusTotal zerinden apraz kontrol edilerek onaylanır. Bylece veri setine dahil edilen yazılım rneklerinin gerekten zararlı olduđu kesinleřtirilir.

rnek sayısının fazlalıđından dolayı bunu otomatik yapmak alıřmayı hızlandırmıřtır. Otomatik sırayla dođrulama yapabilecek bir python programı yazılmıř ve bununla birlikte hızlı bir řekilde yksek oranda dođrulamalar gerekleřtirilmiřtir. Dođrulamadan geemeyen rnekler, setten ıkarılmıřtır. Sayılar izelge 3.1'de gsterilmiřtir.

Ayrıca, alıřmada stratejik olarak referans alınan MOTIF [37] veri seti alıřmaları

da, geniş ve çeşitlilik içeren zararlı yazılım veri setlerinin önemini vurgulamaktadır. Malware Bazaar'dan elde edilen örnekler, MOTIF yaklaşımının temel prensipleriyle uyumlu olarak, sürekli güncellenen, detaylı meta veri içeren ve yüksek doğrulukta zararlı yazılım örnekleri sağlamaktadır. Bu sayede, çalışma kapsamında kullanılan zararlı örneklerin, dinamik analiz ve sınıflandırma süreçlerinde sağlam bir temel oluşturması mümkün hale gelmiştir.

3.1.2 Zararsız Yazılımların Toplanması

Dinamik davranış analizinde, zararlı yazılım örneklerinin yanı sıra, zararsız davranış kalıplarını doğru öğrenebilmek için zararsız yazılım örneklerinin toplanması son derece kritik bir adımdır. Güvenli yazılımlar, genellikle sistem üzerinde pasif ve günlük kullanımda görülen davranışlar sergilediğinden, bu örneklerin toplanması sırasında özellikle dikkatli olunması gerekmektedir. Çünkü, zararsız davranışlar çok düşük etki ve aktif olmayan kalıpları içerdiği için, modelin bu pasif davranışların geniş bir yelpazesini öğrenebilmesi adına farklı özelliklere sahip, çeşitli uygulama kategorilerinden örnekler toplanmalıdır.

Bu çalışma kapsamında, zararsız yazılım örnekleri öncelikle **CNET**, **SourceForge** ve **Softonic** gibi güvenilir ve geniş uygulama portföyüne sahip platformlardan temin edilmiştir.

- **Softonic:** Kullanıcı yorumları, güncel sürüm bilgileri ve geniş yazılım portföyü ile zararsız yazılım örneklerinin çeşitliliğini sağlamada etkili bir kaynaktır [22].
- **SourceForge:** Açık kaynak yazılım geliştirme ve dağıtımına yönelik zengin içeriğiyle, farklı kategorilerden zararsız uygulamaların bulunduğu önemli bir kaynaktır [23].
- **CNET:** Güncel teknoloji ve yazılım incelemelerinin yapıldığı, kullanıcı değerlendirmeleri ve indirme linklerinin sunulduğu önde gelen bir platformdur [24].

Çizelge 3.1: Zararlı Veri Seti.

Toplanan Zararlı Dosya Sayısı	Doğrulan Sayı
3000	2854

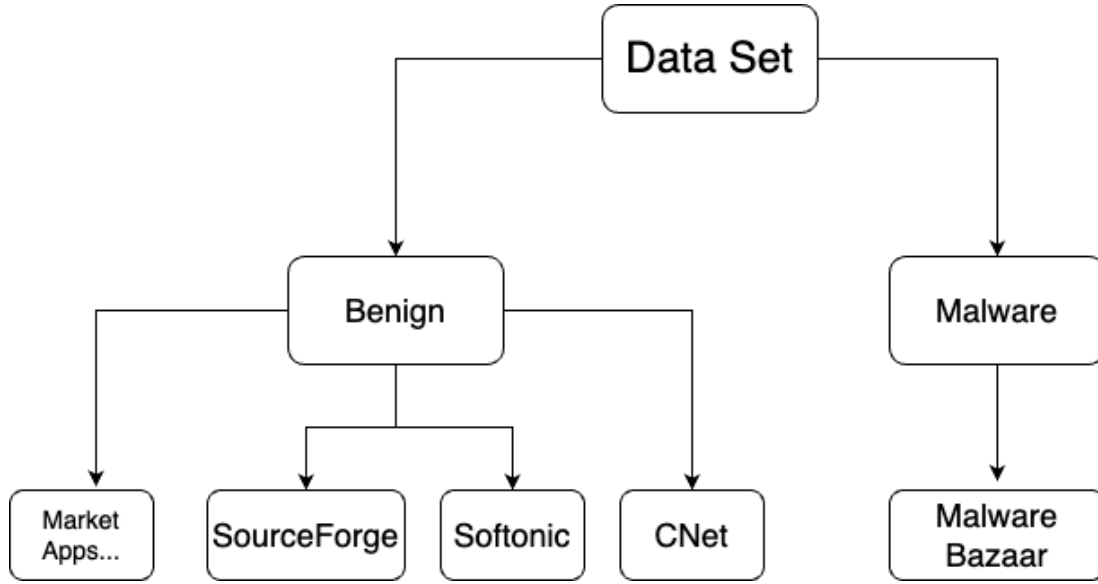
Toplanan her bir zararsız yazılım örneğini zararlı yazılımdaki sistemle aynı olarak otomatik olarak sırayla kontrol edilmiştir. Şüpheli olduğu düşünülen örnekler listeden çıkarılmıştır. Her toplanan platforma göre oluşan sonuçlar Çizelge 3.2’nde gösterilmektedir.

Çizelge 3.2: Zararsız Veri Seti.

	Toplanan Zararsız Dosya Sayısı	Doğrulan Sayı
CNET	1301	1294
SourceForge	3220	3202
Softonic	155	152

Ayrıca, zararsız yazılımların pasif ve çeşitli doğası nedeniyle, modelin normal sistem davranışlarını daha doğru öğrenebilmesi için farklı uygulama kategorilerinden örneklerin toplanması önemlidir. Bu kapsamda, yakın zamanda yayınlanan bir çalışma, Windows çalıştırılabilir dosyalardan oluşan çoklu özellikli zararsız veri setlerinin toplanmasının, dinamik analiz ve model eğitiminde önemli avantajlar sağladığını ortaya koymuştur [38]. Böylece, toplanan güvenilir örneklerin çeşitliliği, sistem kayıtlarında ve API çağrı kalıplarında farklı normal davranışları temsil ederek, sınıflandırma modelinin yanlış pozitif oranlarını düşürmesine ve genel performansın artmasına katkı sağlamaktadır.

Tablodaki rakamlardan, zararsız veri setinde özellikle SourceForge üzerinden yoğun veri temin edildiği gözlemlenmektedir. Bunların dışında, birkaç market uygulamaları; Youtube, güncel oynanan oyunlar(League Of Legends vb...), Photoshop eklendi. Çünkü güncel en çok kullanılan uygulamaları kayıtları da faydalı olacaktır. Ayrıca, zararsız uygulamaların pasif doğası nedeniyle, bu uygulamalardan elde edilecek dosya sayısının zararlı örneklerden daha fazla olması önemlidir. Çünkü yeterli veri toplanmadığında, zararsız davranışların temsilinde eksiklik yaşanabilir. Bu nedenle, çalışmada daha çok zararsız dosya toplandı. Böylelikle Şekil 3.1’deki diyagramdaki görüldüğü gibi zararlı ve zararsız belli sayıda veriler çeşitli yerlerden toplanması sağlandı.



Şekil 3.1: Veri Seti Diyagramı.

3.2 Veri Çalıştırma Süreci

Bu bölümde, dinamik analiz kapsamında, uygulamaların çalışma anındaki davranışlarını izlemek amacıyla gerçekleştirilen veri toplama ve işleme süreçleri detaylandırılmaktadır. Aşağıda, kayıt toplama yöntemleri ile otomatik işlem ve deneysel akışın nasıl yürütüldüğü anlatılmıştır.

3.2.1 Kayıt Toplama Yöntemleri

Dosyaların toplanmasının ardından, çalışmanın verimli olması için hangi verilerin kullanılacağı belirlenmelidir. Dinamik analiz planlandığından, dosyalar çalıştırılırken ortaya çıkan davranışların anlık olarak izlenip kaydedilmesi gerekmektedir. Kaydedilen kayıt kayıtlarının gerçekten işe yarar ve anlamlı bilgiler içerdiğinin doğrulanması kritik öneme sahiptir. Bu süreci, teorik olarak basit bir dosya üzerinde test ederek ilerlemek mümkündür. Fonksiyonel olarak bakıldığında, sistemin önce belirli bir dosyayı seçmesi, ardından o dosyayı önceden tanımlanmış bir süre boyunca çalıştırması ve bu esnada ortaya çıkan davranışların detaylı olarak izlenip kaydedilmesi gerekmektedir. Bu aşamada, çalışmanın ana odak noktası olan Windows olay kayıtları, dosyanın çalıştığı süre boyunca üretilen verilerin toplanması ve saklanması üzerine yoğunlaşmaktadır.

Windows Olay Kayıtları, Windows işletim sisteminde gerçekleşen tüm önemli sistem,

uygulama ve güvenlik olaylarının otomatik olarak kaydedildiği günlük dosyalarıdır. Bu kayıtlar, işletim sisteminin performansını, stabilitesini ve güvenliğini izlemek amacıyla kullanılır. Her bir olay kaydı, olayın meydana geldiği zaman damgası, olay kimliği, olayın kaynağı, olayın türü ve açıklaması gibi detaylı meta veriler içerir.

Windows Olay Kayıtları, farklı amaçlara yönelik birden fazla kanalda toplanan olaylardan oluşur. Örneğin, **Application** kanalı, sistemde yüklü uygulamaların çalışması sırasında meydana gelen olayları; **System** kanalı, işletim sistemi bileşenleriyle ilgili olayları; **Security** kanalı ise güvenlikle ilgili girişimler, denetim olayları ve erişim kontrollerini kaydeder. Buna ek olarak, **Setup**, **Forwarded Events** gibi kanallar da bulunabilir. Her kanal, kendi içerisinde olayın zaman damgası, olay kimliği, kaynak, kategori ve açıklama gibi detaylı meta veriler içerir.

Örneğin, Windows Olay Kayıtlarının uygulama hataları, sistem uyarıları, güvenlik ihlalleri, donanım arızaları ve diğer kritik sistem olaylarını belgeleyerek, sistem yöneticilerine ve güvenlik uzmanlarına, sistemdeki olağan ve olağan dışı davranışlar hakkında bilgi sunar. Dinamik davranışsal analizi bağlamında, bu kayıtlar, zararlı yazılımların çalıştırıldığı ortamda hangi API çağrılarının yapıldığını, hangi dosya ve kaynaklara erişildiğini, ağ bağlantılarının nasıl gerçekleştiğini ve sistem ayarlarında ne tür değişikliklerin yapıldığını ortaya koyar. Böylece, Windows Olay Kayıtları, hem sistemin normal çalışma düzeninin izlenmesinde hem de zararlı davranışların tespitinde kritik bir veri kaynağı olarak öne çıkar.

Algoritma 1: Bir Çalıştırılabilir Dosyanın Kayıtlarının Toplanma Süreci.

Veri: Çalıştırılabilir Dosya

Sonuç: Data'nın Windows kayıtları

Dosyayı seç;

Çalışacak Süreyi Belirle;

Belirlenen süre boyunca dosyayı çalıştır;

Çalışma sırasında Windows Olay Kayıtlarını ve terminal kayıtlarını topla;

Toplanan kayıtları dosyaya kaydet;

Bu pseudocode'u temel alan sistem, Python üzerinde geliştirilecek bir program aracılığıyla gerçekleştirilecektir. Python'da yer alan "win32evtlog" kütüphanesi sayesinde Windows olay kayıtlarına erişilebilmektedir [39]. Program, shell script benzeri bir mekanizma kullanarak çalışır durumda iken, kütüphane aracılığıyla elde edilen her bir

kaydını dosyaya yazarak toplar. Ancak, win32evtlog kütüphanesi kayıtları XML formatında tuttuğundan, makine öğrenmesi işlemleri öncesinde daha kolay işlenebilmesi için bu kayıtlar JSON formatına dönüştürülmektedir. Böylece, model eğitimi için daha esnek ve uygun yapıda veri elde edilmiş olur.

Bu kayıt toplama stratejisi, dinamik analizde her bir uygulamanın davranış kalıplarının ayrıntılı ve doğru şekilde kaydedilmesini sağlar. Böylece, zararsız ve zararlı yazılımlar arasındaki davranışsal farklar daha net ortaya konulabilmekte ve modelin eğitiminde kullanılacak öznitelikler daha sağlam temellere oturtulmaktadır.

Çizelge 3.3 içerisinde görünen dosya kayıtları, çalıştırılan dosyanın her kanaldan oluşan dosya kayıtlarını ifade etmektedir. "**time_log.txt**" dosyaların çalıştırıldığı sürelerin kaydedildiği basit bir text dosyasıdır. Security, System gibi ana kanallardan kayıtların geldiği bu örnekte görülebilir. Dosya içerisindeki kayıt detayları ilerleyen kısımlarda daha detaylı açıklanacak.

Çizelge 3.3: Çalıştırılan Bir Dosyanın Kayıtları.

Dosya / Klasör Adı
logcat_log.txt
Microsoft-Windows-Application-Experience%4Program-Telemetry.evtx.csv
Microsoft-Windows-LiveId%4UX-Notifications%4ActionCenter.evtx.csv
Microsoft-Windows-Security-SPP-UX-Notifications%4ActionCenter.evtx.csv
Microsoft-Windows-SMBServer%4Operational.evtx.csv
Microsoft-Windows-Sysmon%4Operational.evtx.csv
Microsoft-Windows-WER-PayloadHealth%4Operational.evtx.csv
Microsoft-Windows-WMI-Activity%4Operational.evtx.csv
Security.evtx.csv
System.evtx.csv
time_log.txt

3.2.2 Otomatik İşlem ve Deneysel Akış

Bir dosyanın kayıtlarını başarıyla topladıktan sonra, aynı sürecin 7000'in üzerinde farklı dosya için uygulanması gerektiği ortaya çıkmıştır. Her bir dosyayı manuel olarak çalıştırmak ve belirli süre boyunca gözlemleyerek kayıt toplamak, hem zaman hem de insan kaynağı açısından oldukça maliyetli bir yaklaşımdır. Bu durum, dina-

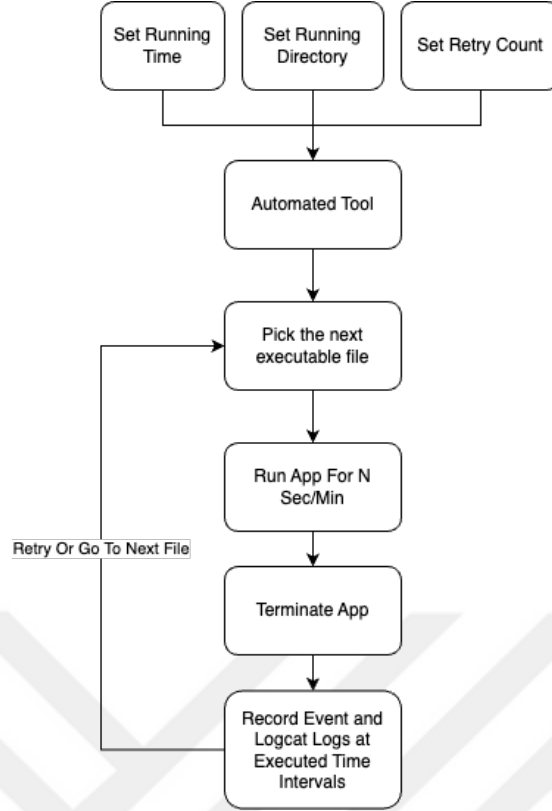
mik analiz sürecinin otomasyonunu zorunlu kılmaktadır. Çünkü dinamik analizde elde edilen kayıtların doğruluğu ve kapsamı, çalıştırma süresi ve işlemlerin düzenli şekilde tekrarlanmasına bağlıdır. Manuel tetikleme ise hatalara, eksik kayıt toplamaya veya süreçte aksamaya neden olabilir.

Bu nedenle, tüm çalıştırılabilir dosyalar tek tek tetikleneceğine, bu dosyaların merkezi bir dizinde toplanması ve programın artık tek bir dosya yerine bu dizini girdi olarak alması planlanmıştır. Böylece, geliştirdiğimiz sistem dizindeki tüm dosyaları sırayla çalıştıracak, her birini önceden tanımlanmış bir süre (saniye veya dakika bazında) çalıştırarak gözlemleyecek ve bu süre boyunca ortaya çıkan Windows olay kayıtlarını toplayacaktır. Bu yaklaşım, hem tutarlılığı sağlar hem de binlerce dosyanın daha kısa zamanda ve aynı kalite standartlarında incelenebilmesine olanak tanır.

Oluşturulan otomasyon yapısı, dinamik analiz sürecinin güvenilirliğini ve tekrarlanabilirliğini artırmakla kalmaz, aynı zamanda modelin eğitiminde kullanılan veri setinin genişlemesine ve çeşitlenmesine de katkı sağlar. Yüksek sayıda dosyanın aynı yöntem ve sürelerde çalıştırılması, her dosyanın benzer koşullarda analiz edildiği anlamına gelir. Bu da elde edilen kayıtların daha bütüncül bir biçimde karşılaştırılabilmesine ve sınıflandırma modelinin genel performansını iyileştirmeye yardımcı olur. Otomasyon süreci, veri toplama işleminin sürekli ve kesintisiz yürütülmesini sağlayarak, dinamik analizde yüksek doğrulukta ve ayrıntılı kayıt verileri elde edilmesine olanak tanır [40]. Şekil 3.2 akışı göstermektedir.

3.3 Sanal Ortam ve İzolasyon Teknikleri

Dinamik analizde, analiz sürecinde kullanılacak dosyaların çalıştırılacağı ortamın izole ve güvenli olması, hem analiz doğruluğunu artırmak hem de çalışma ortamının ağ veya diğer cihazlara zarar vermemesini sağlamak açısından kritik bir öneme sahiptir [12]. Bu amaçla, sanal makine (VM) ortamları tercih edilmekte ve Windows işletim sistemine yönelik olarak özel olarak yapılandırılmış Windows sanal makineleri kullanılmaktadır. Bir sanal makine, fiziksel bir bilgisayarın kaynaklarını paylaşarak çalışan, izole edilmiş ve bağımsız bir çalışma ortamı sunan yazılımsal bir sistemdir. Bu ortam, gerçek donanıma benzer öznitelikler sergilerken, test, analiz ve geliştirme işlemleri sırasında fiziksel sistemin korunmasını sağlar. Çalışmanın ihtiyaçlarına göre sanal ma-



Şekil 3.2: Deneysel Akış.

kine kurulumu dikkatlice planlanmalıdır. Çalışma kapsamında çok sayıda dosyanın otomatik olarak çalıştırılması gerekeceğinden, sanal makinenin yüksek performanslı donanım özelliklerine sahip olması beklenir. Program setimiz Windows dosyalarını içerdiğinden, sanal makineye Windows işletim sistemi kurulmalı ve .NET Framework gibi gerekli eklenti ve araçların da yüklü olduğundan emin olunmalıdır. Bunun ötesinde, en kritik husus sanal makinenin zararlı davranışlar tarafından tespit edilebilirliğidir. Günümüzde, zararlı yazılımlar sanal ortamların varlığını tespit edebilecek algoritmalar kullanabilmektedir. Zararlı yazılımlar, çalışmaya başlamadan önce bulunduğu ortamı analiz eder. Eğer ortamın analiz şüphesi varsa, tehlikeli davranışlarını gizlemek veya değiştirmek için farklı yöntemlere başvurur. Bu durum, dinamik analiz sürecinde elde edilen verilerin doğruluğunu ve güvenilirliğini doğrudan etkiler [41].

3.3.1 Kaynak Tahsisi

Sanal makine, gerçek bir fiziksel cihazın performansını taklit edebilmek için yeterli kaynaklarla donatılmalıdır. Düşük kaynak kullanımı, yazılımın sanal ortamda veya

analizde olduğunu anlayabileceği ipuçlarından biridir. O yüzden yüksek bellek ve işlemci kaynakları ortam olarak gereklidir. Bu çalışmada, VM'ye minimum 100 GB depolama alanı (SSD veya HDD), 16 GB RAM ve 4 çekirdekli işlemci atanmıştır. Bu kaynak tahsisi, sanal makinenin modern fiziksel makinelerle benzer performans göstermesini sağlayarak, zararlı yazılım tarafından tespit edilebilecek performans sapmalarını minimize eder. Sandborn ve arkadaşları (2024) [42] çalışmalarında, sanal makinelere yetersiz donanım kaynaklarının özellikle CPU ve bellek kısıtlamaları nedeniyle belirgin performans düşüşlerine yol açtığı, bu düşüşlerin ise gelişmiş zararlı yazılım tarafından sanal ortam tespiti için kullanılabildiği ortaya konulmuştur. Dolayısıyla, yeterli donanım kaynaklarının atanması, sanallaştırma katmanının getirdiği overhead'in maskelenmesi ve analiz ortamının gerçek bir fiziksel sistem izlenimi vermesi açısından kritik bir önlemdir.

3.3.2 Donanım İzi Gizleme

Agresif davranışlı programlar, sanal ortamlarda çalıştıklarını tespit etmek için donanım imzalarını kullanabilir. Bu nedenle, sanal makinedeki MAC adresleri, CPU model isimleri ve BIOS bilgileri gibi donanım tanımlayıcıları, fiziksel makineleri andıracak şekilde modifiye edilmesi gerekir. Böylece, zararlı yazılımlarının sanal ortamı tespit etme ihtimali azaltılarak, analiz sırasında gerçek davranışlarının ortaya çıkması hedeflenir [42].

3.3.3 İzole Edilmiş Sanal Ağ

Sanal makinenin oluşturulduğu izole ağ ortamında, dış dünyaya veya bağlı diğer cihazlara zarar vermemesi için özel bir ağ yapılandırması uygulanmıştır. VirtualBox gibi sanallaştırma araçları kullanılarak, yalnızca yerel iletişime izin veren, iç ağ modunda çalışan bir ortam oluşturulmuştur. Bu yapı sayesinde, analiz sırasında çalıştırılan dosyaların gerçek ağ ortamına erişimi engellenmiş, kötü niyetli aktivitelerin sistem dışına sızması önlenmiştir. Böylece, toplanan kayıt verilerinin güvenilirliği artarken, analiz ortamı dış etkenlerden izole edilmiş ve güvenli bir çalışma zemini sağlanmıştır.

İzole edilmiş bu ağ ortamında, dinamik analizde ortaya çıkan ağ trafiği ve diğer davran-

nıssal verilerin detaylı incelenebilmesi için **Remnux** kullanılmıştır. Remnux, dinamik analizde ve tersine mühendislik çalışmalarında yaygın olarak tercih edilen, kapsamlı bir sanal ortam sunar. Ağ sistemini sanki gerçekmiş gibi simüle etmeye çalışır. Bu sistem ile, dosya analizi, hafıza forenzik çalışmaları ve ağ trafiği izleme gibi görevleri entegre bir biçimde gerçekleştirerek, yazılımların analiz ortamında ürettiği verilerin detaylı ve güvenilir bir şekilde toplanmasına olanak tanır. Özellikle ağ izleme çalışmalarında kullanılır [43]. Remnux, fakenet ve inetsim komutları ile bu yetenekleri sağlar.

Fakedns, DNS sunucusu davranışını simüle eden bir komuttur. Zararlı yazılım, komuta kontrol sunucularına veya diğer hedeflere ulaşmak için DNS sorguları gönderir. "fakedns", bu sorgulara sahte yanıtlar vererek zararlı yazılımların gerçek DNS sunucularına erişimini engeller. Bu sayede, analiz ortamı dışına istenmeyen veri sızıntısı önlenirken, zararlı yazılımların DNS sorguları üzerinden gerçekleştirdiği iletişim davranışları da detaylı olarak incelenebilir.

Inetsim ise, çeşitli internet servislerini simüle eden bir komuttur. Inetsim, HTTP, FTP, SMTP gibi protokolleri taklit ederek, zararlı yazılımların bu servislere yaptığı bağlantı denemelerini kontrol altında tutar. Yazılımlar genellikle gerçek bir sunucu ile iletişime geçmeye çalıştığında, inetsim sayesinde bu iletişim izole edilmiş ve kontrollü bir ortamda gerçekleşir.

3.3.4 Ortam Göstergelerinin Gizlenmesi

Zararlı yazılımlar, sanal ortamda çalıştıklarını tespit edebilecek belirli dosya adları, işlem isimleri ve sistem göstergelerine dayanarak analiz ortamını ayırt edebilmektedir. Bu nedenle, sanal makinelere özgü standart işlem isimleri ("VirtualMachineProgram-NameService.exe") kasıtlı olarak değiştirilmekte, sistem dosyaları ve işlem tanımlayıcıları farklı adlarla maskelenerek yazılımların bu tür belirgin kalıpları kullanarak sanal ortamı tespit etme olasılığı azaltılmaktadır. Bu tür modifikasyonlar, analiz sürecinde elde edilen kayıt verilerinin doğruluğunu korumak ve tehlikeli programların kaçınma tekniklerini etkisiz hale getirmek açısından kritik bir öneme sahiptir [44].

3.3.5 Yazılım Göstergeleri

Sanal makine, gerçek bir fiziksel sistem izlenimi vermesi amacıyla, ek yazılımlar ile desteklenmiştir. Ortamı daha doğal gösterebilmek için Adobe Reader, Google Chrome, Mozilla Firefox gibi yaygın ofis ve internet uygulamalarının yanı sıra, VLC Media Player, Skype, Discord gibi sosyal ve eğlence amaçlı uygulamalar da kurulmuştur. Bu ek uygulamalar, yalnızca kullanıcı arayüzü sağlamakla kalmayıp, sistemin arka planda gerçekleştirilen süreçlerine, API çağrılarına ve dosya erişim aktivitelerine de yansıtılarak, analiz sırasında yazılımın ortamı tespit etme çabalarını zorlaştırmaktadır. Ayrıca, WinRAR veya 7-Zip gibi genel amaçlı yardımcı yazılımlar, dosya sıkıştırma, arşivleme ve dosya yönetimi gibi işlemleri gerçekleştirirken, sistemin normal kullanım senaryolarını yansıtan çeşitli aktiviteleri tetikler. Bu sayede, sanal makinede oluşan kayıt verilerinde, gerçek bir kullanıcının sistemde sergilediği davranışlar daha net bir şekilde izlenebilir ve zararlı yazılımın bu davranışları tespit etmesi zorlaşır. Böylece, analiz ortamında elde edilen verilerin doğruluğu ve güvenilirliği artar. Microsoft Office uygulamalarının kurulması ise ortamın gerçek bir kullanıcı sistemi izlenimini güçlendirmenin yanı sıra, zararlı yazılımın Office dosyalarına yönelik saldırı davranışlarının kaydedilebilmesine olanak tanır. Office uygulamaları, kullanıcıların günlük yaşamlarında yoğun olarak kullandığı programlar olarak, sistemde tipik aktiviteleri ve veri erişim örüntülerini oluşturur. Bu aktivitelerin varlığı, analiz sırasında normal sistem davranışlarının modellenmesine yardımcı olurken, yazılımların Office uygulamaları üzerinden gerçekleştirebileceği makro saldırıları gibi kötü niyetli aktivitelerin de izlenmesini mümkün kılar. Böylece, sanal makine ortamı, gerçek dünyadaki bir kullanıcı sistemini andıran ve yazılım türü analizinde kritik verilerin doğru şekilde toplanmasına yardımcı olan sağlam bir platform haline gelir.

3.3.6 Ek Ağ ve Sistem Araçları

Olay kayıtlarının kapsamını genişletmek amacıyla, yalnızca sistem kayıtlarının değil, aynı zamanda ağ trafiği kayıtlarının da toplanabilmesi için Npcap gibi araçlar kullanılmıştır. Sysmon gibi araçlar, Windows Olay Kayıtlarının sağladığı temel bilgilerin ötesinde, işlem oluşturma, dosya oluşturma, ağ bağlantısı türü ve diğer kritik sistem aktivi-

telerini detaylı şekilde kaydederek kayıt kapsamını artırmaktadır. Ancak, zararlı yazılımlar Sysmon gibi ek kayıt toplama araçlarını tespit edebilmektedir. Bu nedenle, analiz ortamında farklı senaryoları değerlendirerek hem Sysmon'un kurulu olduğu hem de kurulu olmadığı durumlarda kayıt toplama işlemi gerçekleştirilmiştir. Bu yöntem, ortamın spesifik yapılandırmasına aşırı bağımlı kalmadan, genel ve esnek bir kayıt toplama süreci sunarak, zararlı yazılımın saklanma stratejilerine karşı daha dirençli bir analiz gerçekleştirilmesine olanak tanımaktadır. Sysmon, Windows işletim sisteminde daha derinlemesine ve ayrıntılı kayıt verileri elde etmek için kullanılan bir araçtır. Örneğin, varsayılan Windows olay kayıtları yalnızca temel sistem olaylarını kaydederken, Sysmon; process, ağ, dosya ve hatta zaman damgası gibi detayları da içeren geniş kapsamlı kayıtlar üretir. Bu ek bilgiler, yazılımların davranışlarının analizi açısından kritik önem taşırken, agresif davranışlı programların bu ek kayıtları tespit etme riski de göz önünde bulundurularak, ortamda farklı konfigürasyonlar test edilmiştir. Böylece, analiz süreci, Sysmon'un sağladığı derinlemesine kayıt verilerinden faydalanırken, zararlı yazılımın potansiyel saklanma tekniklerine karşı da esnek bir yaklaşım sergilemektedir. Özetle, Npcap, Sysmon gibi araçların entegrasyonu sayesinde, izole edilmiş ağ ortamında elde edilen kayıt verileri hem daha detaylı hem de güvenilir hale getirilmektedir. Bu durum, zararlı yazılım analizinde, kötü niyetli aktivitelerin doğru şekilde tespit edilmesi ve analiz sonuçlarının yüksek doğrulukla değerlendirilmesi açısından büyük önem taşımaktadır [45].

3.4 Deneysel Kurulum ve Çalışma Akışı

Mikro mekanizmaların kurulumu tamamlanmış ve ihtiyaçlara göre değerlendirilmiştir. Sandbox ortamının çalıştırılabilir dosya döngüsüyle entegrasyonu aşamasına gelindiğinde, bu entegrasyon sırasında yaşanabilecek olumsuzlukların ve uyumsuzlukların dikkatle izlenmesi gerekmektedir. Çünkü zararlı yazılım çalıştırmak basit bir işlem olmamakla birlikte, %100 zararlı yazılımın sanal ortamdan kaçışını engellemek de gerçekçi bir beklenti değildir. Zararlı yazılımlar, çalıştığı sanal makineyi işlevsiz bırakmaya çalışabilir, tüm işlemi aniden sıfırlayabilir veya sanal makineyi yeniden başlatmaya zorlayarak, arka planda kurulumun izlerini taşıyan kayıt toplama sürecini aksatabilir. Hatta bazı durumlarda, yazılım yalnızca otomatik sistemi değil, aynı zamanda tüm dosyaları hedef alarak, toplanan kayıt verilerini veya çalıştırılacak veri setlerini

bozabilir [16]. İlk adım olarak, bir sanal makine sandbox'ı kurulur ve bu kurulum, önceki başlıklarda belirtildiği gibi gerekli şartlara uygun şekilde yapılandırılır. Toplanan dosya seti, sandbox ortamına toplu olarak yüklenir. Ardından, otomatik çalışmayı sağlayan Python tabanlı bir program, Windows'a uygun olarak exe formatında derlenir ve sandbox ortamına aktarılır. Kurulum sürecinde, sandbox'ın tüm gerekli uygulamaları, ek araçları ve yardımcı yazılımları (örneğin, .NET Framework, gerekli dosya uzantı araçları vb.) yüklenerek, zararlı yazılım analizinde ihtiyaç duyulan ortamın tam olarak oluşturulması sağlanır.

3.4.1 İlk İzlenim

İlk çalıştırma deneyimleri sırasında, zararsız yazılımların genel olarak stabil çalıştığı gözlemlenmiştir. Ancak, bazı güvenli dosyalar, çalıştırılmaları sırasında beklenmedik şekilde yeniden başlatma sorunları yaşamakta ve sistemin bazı ek araçlara ihtiyaç duyduğunu göstermiştir. Bu durum, erken aşamalarda zararsız örneklerin, büyük çapta bir çökme veya dosya bütünlüğünün bozulması olmaksızın, yalnızca bazı kesintiler ve cihazı yeniden başlatma sorunlarıyla karşılaşıldığını ortaya koymaktadır.

Ayrıca, zararsız uygulamalarda bazı uyumsuzluklar da belirlenmiştir. Çalışma sırasında, belirli dosyaların düzenli aralıklarla yeniden başlatılması gerektiği ve bazı uygulamaların ek yazılım desteği gerektirdiği tespit edilmiştir. Bu gözlemler, analiz ortamının zararsız yazılım tarafında genel olarak stabil kaldığını, fakat bazı durumlarda konfigürasyonun ve ek araçların dikkatle entegre edilmesinin gerekliliğini göstermektedir.

Tehlikeli dosyaların çalıştırılması esnasında ise, zararsız uygulamalara göre çok daha ciddi sorunlar gözlemlenmiştir. Bazı yazılım örnekleri, çalıştırılmaları sırasında kendi bileşenlerini bozarak aniden sistemin yeniden başlatılmasına yol açmış, bu da yazılımların saklanma stratejilerinin bir parçası olarak ortaya çıkmıştır. Bu tür beklenmedik yeniden başlatmalar, zararlı yazılımların analiz ortamında gerçek davranışlarını gizlemek amacıyla sergilediği taktiklerden biridir.

Yeniden başlatma sonrasında, görev yöneticisinde bilinmeyen ve beklenmedik işlem isimleriyle çoğalan süreçler tespit edilmiştir. Bu durum, zararlı yazılımların arka planda

gizli işlemler başlatarak kendi faaliyetlerini maskeleymeye çalıştığının göstergesidir. Programlar, bu şekilde analiz ortamının tespit edilmesini zorlaştırırken, aynı zamanda sistem kaynaklarını tüketerek analiz sürecinde belirsizliklere neden olmaktadır.

Daha ileri aşamalarda, bazı zararlı yazılım örneklerinin yalnızca kendi çalışma süreçlerini bozmakla kalmadığı, aynı zamanda kayıt dosyaları da dahil olmak üzere tüm dosya sistemini etkilediği gözlemlenmiştir. Agresif programlar, analiz ortamının bütünlüğünü bozarak, toplanan kayıt verilerinin doğruluğunu azaltmak ve sonuçları yanıltmak amacıyla agresif stratejiler uygulamaktadır. Bu tür davranışlar, dinamik analiz sürecinde ciddi veri kayıplarına ve yanlış sonuçlara yol açabilmektedir.

Tüm dosya tiplerinin ardışık çalıştırılması sırasında, her bir dosyadan elde edilen kayıtların boyutunda kademeli bir azalma gözlemlenmiştir. Bu azalma, önceki dosyalardan kalan verilerin yeni dosya kayıtlarına eklenmesi sonucu ortaya çıkan birikimden kaynaklanmaktadır. Sonuç olarak, her dosya kendi özgün kayıtlarını tam anlamıyla oluşturamayarak, analiz sürecinde toplanan verilerin netliğini ve tekrarlanabilirliğini olumsuz yönde etkilemektedir.

Çözüm olarak, ilk adım zararsız ve zararlı dosyaların çalıştırılması sürecini ayırt etmekten başlamıştır. Bu aşamada, ekstra bir sanal makine üretilerek aynı sandbox koşulları altında güvenli ve zararlı dosyalar ayrı ayrı çalıştırılmıştır. Güvenli dosyaların çalıştırılması sırasında toplanan kayıtlar, zararlı dosyaların yarattığı sistem bozulmalarından ve kayıtların zarar görmesinden korunmuş ve bu sayede, zararsız dosya kayıtları, zararlı yazılımların neden olduğu veri kayıplarının önüne geçerek analiz sürecinde kullanılmak üzere sağlam bir referans oluşturmada önemli rol oynamıştır. Örneğin, zararsız yazılım kayıtlarının izole edilmiş ortamda düzgün şekilde toplanması, zararlı dosyalardan kaynaklanan aşırı hata ve bozulma durumlarında bile normal sistem davranışlarının net bir şekilde gözlemlenmesine olanak tanımıştır.

Buna ek olarak, zararsız yazılım dosyalarının ardışık olarak çalıştırılması sonucunda her çalıştırmada kayıt boyutlarında artış durumu devam etmiştir. Diğer yandan sandboxları ayırma, zararlı dosyaların çalıştırılması esnasında ise oluşan problemleri senaryoları çözmemiştir. Agresif davranışlar, sistem kaynaklarını tüketmekle kalmayıp, toplanan kayıtların bütünlüğünü de olumsuz yönde etkilemiştir. Sonuç olarak, aynı sandbox

ortamında ardışık olarak zararsız ve zararlı dosyaların çalıştırılması, her iki tarafın da kendine özgü etkilerinin ortaya çıkmasına neden olmuş. Zararsız yazılım kayıtları, zararlı yazılımların zarar verici etkilerinden korunmuş olsa da, her yeni çalıştırma ile kayıt boyutlarında artış gözlemlenmiş, böylece analiz sürecinde her dosyanın kayıtlarının tekiliği ve netliği azalabilmiştir. Genel olarak sorunlar Çizelge 3.4 içerisinde özetlenmiştir.

Çizelge 3.4: İlk İzlenimde Zararsız ve Zararlı Dosyaların Oluşturduğu Sorunlar.

Güvenli Dosyalarda Gözlemlenen Sorunlar	Zararlı Dosyalarda Gözlemlenen Sorunlar
• Beklenmedik yeniden başlatma sorunları • Ek araç ve güncelleme gereksinimi • Uyum sorunları ve bazı dosyaların ardışık çalıştırmada hata üretmesi • Ardışık çalıştırmalarda olay kayıt boyutlarında kademeli artış	• Zararlı program, çalıştırma sırasında belirli bileşenleri bozarak sistemi yeniden başlatıyor • Görev yöneticisinde bilinmeyen işlem isimlerinin çoğalması • Kayıt dosyalarının veya tüm dosya sisteminin bozulması • Tehlikeli programın arka planda sürekli çalışarak kaynakları tüketmesi

Zararsız yazılımların çalıştırılması sırasında, yeniden başlatma sorunlarının çözümünden ziyade, bu durumlara adapte olunması gerektiği gözlemlenmiştir. Çünkü, bir uygulamanın yeniden başlatma öncesi ve sonrası davranışları farklılık gösterebilir. Bu nedenle, Python tabanlı otomasyon programında, her dosya için yeniden başlatma senaryosunu yakından takip edecek ve dosyanın yeniden başlatma öncesi ile sonrası kayıtlarını karşılaştırarak adapte olacaktır. Ayrıca, döngünün çok uzamaması için, her dosyanın yalnızca bir veya iki kez yeniden başlatılmasına izin verilecek, böylece sistemde oluşabilecek gereksiz gecikmeler önlenecektir. Eğer yeniden başlatma sırasında kayıtların kaybolması söz konusu olursa, dosyanın çalıştırıldığı belirli zaman aralıkları önceden kaydedildiğinden, bu aralıktaki Windows olay kayıtları çıkarılabilecektir. Böylece, davranışların izlenmesi daha verimli hale gelecektir.

Ek olarak, sandbox ortamında bulunan ek araçların ve güncellemelerin deneme sayıları arttıkça, sandbox daha stabil bir hale gelmeye başlamaktadır. Bu nedenle, ortam

tam olarak stabil hale gelene kadar, sandbox'da ihtiyaç duyulan program ve araçlar yakından izlenmeli ve sistem bu doğrultuda adapte edilmelidir.

Zararlı dosyalarda ise, yeniden başlatma sorunları için benzer bir yol izlenebilmekle birlikte, yazılımların yarattığı sorunlu senaryolar genellikle sandbox ortamında kalıcı hasara yol açmaktadır. Zararlı yazılım, çalıştırıldığı ortamı bozarak yeniden başlatma veya benzeri müdahaleler sonucunda, sandbox üzerinde kalıcı izler bırakabilir. Bu durum, döngü içerisinde dosyaların sırayla çalıştırılmasıyla elde edilen zararsız yazılımlara özel çözümlerin yeterli olmamasına neden olur. Dosya sayısının çokluğu göz önüne alındığında, manuel müdahale büyük ölçüde zorlaşmış olup, artık sandbox yönetimi konusunda otomatik aksiyonlar alınması gerekmektedir [46].

Ardışık dosyaların çalıştırılmasından elde edilen kayıtların boyutundaki artış ve zararlı yazılımın genel sorunlarına çözüm üretebilmek için, sanal makine sandbox'ı üzerinde senaryolaştırma çalışması yapılması gerekmektedir. Her dosya çalıştırıldığında ortamın eşit kalması, çalıştırma sonrası da aynı temelde devam etmesi hedeflenmelidir. Bu doğrultuda, sanal makinenin durumunu her dosya çalıştırdıktan sonra sıfırlayarak, sonraki dosyanın aynı koşullarda çalıştırılmasını sağlamak için sanal makineye girdi verilerek otomatik bir çalışma döngüsü oluşturulması gerekmektedir.

Bu işlemi mümkün kılan teknoloji, sanal makinelerde yer alan anlık görüntü (Snapshot) teknolojisidir. Anlık görüntü, sanal makinenin o anki durumunun (bellek, CPU, disk, ayarlar) tam bir kopyasını oluşturur ve bu kopya sayesinde, sanal makine üzerinde yapılan değişiklikler hızlı bir şekilde geri alınarak, temiz bir çalışma ortamı elde edilir. Böylece, her dosya çalıştırdıktan sonra ortam eski, başlangıçtaki haline döndürülür. Bu da ardışık dosya çalıştırmalarında eşit koşulların sağlanmasına ve analiz sonuçlarının karşılaştırılabilir olmasına olanak tanır.

Sanal makinede anlık görüntü almanın otomasyonu, dinamik analizde tutarlı bir çalışma ortamı sağlamak için kritik öneme sahiptir. Bu amaçla, VirtualBox'un komut satırı aracı olan "vboxmanage" kullanılarak, sanal makinenin mevcut durumunun tam bir kopyası oluşturulmaktadır [47]. "vboxmanage" komutu ile alınan anlık görüntü, sanal makinenin bellek, disk ve sistem ayarlarını içerdiğinden, herhangi bir müdahale sonrası temiz bir başlangıç sağlar. Her dosya çalıştırıldığında, sistemde meydana ge-

len deęişikliklerin kaybolmaması ve sonraki analiz için ortamın eşit koşullarda sunulabilmesi adına, belirli aralıklarla otomatik olarak anlık görüntü oluşturulmaktadır. Bu işlem, "vboxmanage snapshot <makine_ismi> take" komutu ile gerçekleştirilmekte olup, oluşturulan anlık görüntüler, "vboxmanage snapshot <makine_ismi> restore <anlık_görüntü_ismi>" komutu ile geri yüklenerek sistemin başlangıç durumuna dönmesi sağlanmaktadır. Bu otomatik süreç, analiz sırasında toplanan kayıt verilerinin karışmaması ve her dosyanın bağımsız bir çalışma ortamında incelenebilmesi açısından büyük önem taşır. Ayrıca, vboxmanage komutları komut dosyaları içerisine entegre edilerek, süreç tamamen otomatikleştirilebilmekte ve manuel müdahale ihtiyacı en aza indirilmektedir. Bu sayede, sanal makinede anlık görüntü yönetimi, sürekli ve tutarlı bir analiz ortamı sunarak, yazılım analizinde elde edilen verilerin güvenilirliğini artırmaktadır.

Python ile otomatik çalışacak ve sanal makinenin anlık görüntülerini yönetecek bir program geliştirilmiştir. Bu programda, sanal makine ismi, hazırlanan anlık görüntü ve anlık görüntünün geri yükleneceği durumlar girdi parametreleri olarak verilmektedir.

Sanal makine üzerinde anlık görüntü alınırken dikkat edilmesi gereken bazı hususlar bulunmaktadır. Öncelikle, anlık görüntü, cihazda işletim sistemi tamamen açılmış ve stabil bir durumda iken alınmalıdır. Ayrıca, anlık görüntü alındıktan hemen sonra, dosyaların çok hızlı bir şekilde çalışmaya başlaması gerekmektedir. Bu hızı sağlamak amacıyla bat dosyası çözümü uygulanabilir. Python programının kısayolu oluşturulur ve dosyaların çalıştırılma süreleri kısayola girdi olarak eklenir. Bat dosyası içerisine, kısayol dosyasını çalıştıracak terminal komutları yazılır. Anlık görüntünün cihazdan etkilenmemesi için belirli bir erteleme süresi (10 saniye) eklenir. Program, anlık görüntü alındıktan sonra 10 saniye içinde çalıştırılmaya başlanmalı ve geri yükleme işlemi için 2 dakika süre tanınmalıdır. Tam bat dosyası çalıştırılır iken anlık görüntü kaydedilir. Bu anlık görüntü de artık vboxmanage aracılığı ile açılıp kapatıldıktan sonra tekrar sıfırlanacak şekilde yönetilir. Böylece, her dosya çalıştırıldıktan sonra sanal makine temiz bir duruma geri döndürülerek, sonraki dosyanın aynı başlangıç koşullarında çalışması sağlanır. Algoritma 2 ve 3 içerisinde genel akış görülebilir.

Algoritma 2: Örnek BAT Dosyası Kodu.

Veri: Python kısayol dosyası, Çalıştırılacak dosya listesi, Çalışma süresi

Sonuç: Her dosya çalıştırıldığında gerekli kayıtların toplanması ve makinenin temiz duruma geri döndürülmesi

1. 5 Saniye Bekle;

3. Otomatik Dosya Çalıştıran Programın Kısayoluna girdi olarak dosya listesinin dizinin, çalışma süresini ver ve Çalıştır;

Algoritma 3: VM Anlık Görüntü Yönetimi ile Dosya Çalıştırma Süreci.

Veri: Makine ismi, Anlık görüntü ismi, Dosya klasörü, Çalışma süresi, Geri yükleme süresi

Sonuç: Her dosya çalıştırıldıktan sonra kayıtların toplanması ve sanal makinenin temiz duruma geri döndürülmesi

Adım 1: Sanal makinede otomatik tüm dosyaların üzerinde geçen Python exe programının kısayolunu alıp bat dosyasından 5 saniye sonra çalıştırılabilir hale getir;

Adım 2: Bat dosyasını çalıştır, 5 saniye süre içerisinde sanal makinede anlık görüntü al (vboxmanage snapshot <makine_ismi> take <AnlikGoruntu_ismi>);

her dosya in Dosya Sayısı ,

Adım 3: Anlık görüntü üzerinden sanal makine açılır ve bat dosyası çoktan çalışmaya başlamış olur);

Adım 4: 5 saniye içerisinde Python kısayolu çalıştırılır;

Adım 5: Dosya, belirlenen süre boyunca çalıştırılarak kayıtlar toplanır;

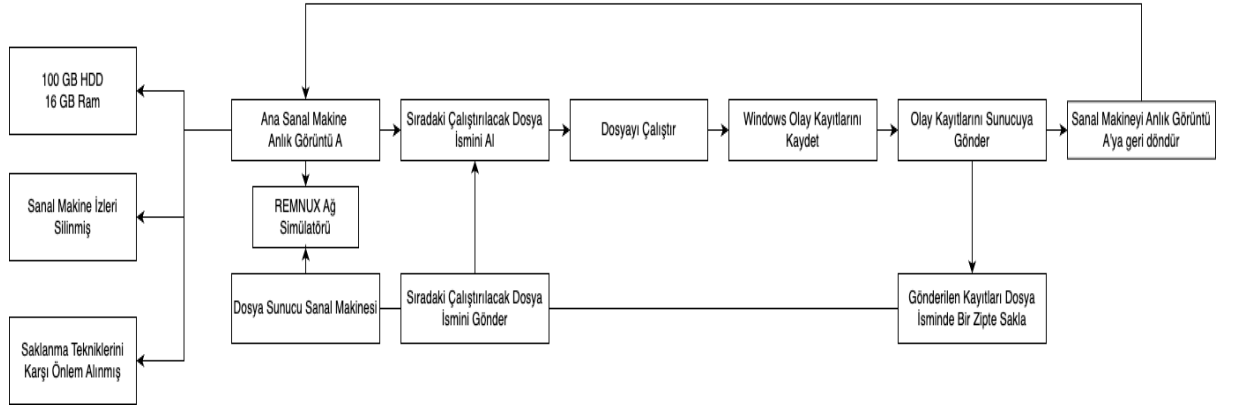
Adım 6: Geri yükleme süresi biter;

Adım 7: Sanal makine kapatılır;

Adım 8: Sanal makine anlık görüntüdeki haline geri getirilir;

Ancak, bu süreçte iki temel sorun öne çıkmaktadır: Birincisi, sanal makine sıfırlandığında sonraki dosyanın ne olduğunu takip edememe, ikincisi ise, zararlı dosyaların çalıştırılması sırasında dosya ve sistem bozulmalarına yol açarak, olay kayıtlarının kaybolmasıdır. Bu ihtiyaçlar göz önünde bulundurulduğunda, dosyalar çalıştırıldıktan sonra, anlık görüntü devreye girmeden önce kayıtlar sanal makine dışındaki izole bir depolama alanına aktarılabilceği bir sisteme ihtiyaç duyulmaktadır. Bu aktarım sürecinde, sanal makineler arası iletişimin verimli ve güvenilir olması kritik öneme sahiptir [48]. Eğer kayıtlar, aynı sanal makine içerisinde saklanırsa, geri yükleme işlemi sırasında bozulabilir ya da tamamen kaybolabilir. Bu da, her dosyanın çalıştırılmasıyla

elde edilen verilerinin tutarlılığını olumsuz etkiler. Bu nedenle, ek bir sanal makine sunucu göreviyle yapılandırılarak, dosyaların çalıştırılmasından elde edilen verilerin merkezi, izole ve düzenli bir şekilde toplanması sağlanmalıdır. Böylece, dosyalar çalıştırıldıktan sonra kayıtlar ayrı bir depolama alanına aktarılır, sistemin bozulma riski minimize edilir ve her çalıştırma için temiz, bağımsız kayıtlar elde edilmiş olur [48]. Python kullanılarak, ayrı bir sanal makine üzerinde sunucu kurulumu gerçekleştirilir. Test sanal makinesi, çalıştırılmadan önce bu sunucuya hangi dosyanın çalıştırılacağı bilgisini sormakta ve sunucudan aldığı cevaba göre ilgili dosyayı çalıştırarak, ürettiği olay kayıtlarını sunucuya göndermektedir. Sunucu, aldığı verileri belirlenmiş bir dizinde saklamakta, böylece her dosyanın kaydı merkezi bir depoda izole ve güvenli bir şekilde toplanmaktadır. Test sanal makinesi, dosya çalıştırıldıktan sonra anlık görüntüye (snapshot) sıfırlanır. Bu işlem, ortamın temiz başlangıç durumuna geri dönmesini sağlar. Sıfırlama tamamlandıktan sonra test sanal makinesi tekrar sunucuya hangi dosyanın çalıştırılacağını sorar. Sunucu, henüz olay kaydı sunucuya aktarılmamış olan ilk dosyayı seçip bilgisini iletir. Böylece, her dosya için kayıtlar ayrı ayrı, eksiksiz ve izole bir şekilde toplanmış olur. İletişim, Remnux üzerinden gerçekleştirildiği için, bu süreç veri güvenliği ve ortam izolasyonu açısından herhangi bir tehlike oluşturmaz. Şekil 3.3 içerisinde tüm akışlar tamamlanmış bir şekilde görünmektedir.



Şekil 3.3: Tamamlanmış Deneysel Akış.

3.4.2 İkinci İzlenim

Yapılan iyileştirmeler ve çözüm önerileri sonrasında, otomatik dosya çalıştırma ve kayıt toplama süreci büyük ölçüde başarılı şekilde ilerlemiştir. Toplamda 7.000'den fazla

zararlı ve zararsız dosyanın analizinde, yalnızca 5–6 kez sanal makineler kendi içlerinde çökme yaşamış, ancak bu durumlarda hızlı bir manuel müdahaleyle süreç kesintiye uğramadan devam edebilmiştir. Bu sayede, otomatik akış büyük oranda sorunsuz biçimde işletilerek, fonksiyonel problemlerin önemli bir kısmı giderilmiştir.

Bununla birlikte, kayıtların en verimli biçimde alınabilmesi için ek adımlar üzerinde yoğunlaşmıştır. Zararlı dosyaların, analiz ortamından kaçınma stratejileri kapsamında, davranışlarını bilerek geciktirdiği veya sakladığı gözlemlenmiştir. Bu nedenle, her dosyanın farklı sürelerde çalıştırılarak toplanan kayıtların değişimini izlemek, en iyi senaryoyu tespit etmek açısından kritik hale gelmiştir. Bu amaçla, 6, 12, 18 ve 25 saniyelik test senaryoları uygulanmıştır. İlk üç sürede (6, 12, 18 saniye) belirgin bir farklılık ortaya çıkmazken, 25 saniyelik çalıştırmalarda **katlanarak** üstel bir artış gözlemlenmiştir. Bu artış, zararlı dosyaların davranışlarını geciktirme stratejilerine karşı daha uzun süreli analiz yapılmasının etkili olduğunu göstermekle birlikte 25 saniyenin en optimal süre olduğunu da belirtir [49].

Şekil 3.4 içerisinde de görüldüğü gibi özellikle zararlı dosyaların ilerlemesinin belli eşikten sonra çok hızlı arttığı fakat zararsızların normal bir ilerlemede olduğu görülmektedir.

Bunun dışında, kaçınmayı etkileyen birkaç önemli faktör daha bulunmaktadır. Örneğin, Sysmon, Windows işletim sisteminde meydana gelen olayların detaylı kaydını tutarak, dosya erişimleri ve ağ bağlantıları gibi kritik aktiviteleri tutmaktadır. Bu sayede, sistemin normal davranışları hakkında daha ayrıntılı bilgi elde edilmektedir. Ancak, işletim sisteminde Sysmon'un bulunması, bazı zararlı örneklerinin analiz ortamını tespit etmesine neden olabileceği gibi, tehlikeli davranışlarını gizleme veya değiştirme stratejilerini tetikleyebilmektedir. Benzer şekilde, Wireshark gibi ağ analiz araçları da,

	Zararsız Dosya Ortalama Kayıt Boyutu	Zararlı Dosya Ortalama Kayıt Boyutu
6 Saniye	92.3 KB	69.7 KB
25 Saniye	0.28 MB	1.6 MB

Şekil 3.4: Sürelere Göre Kayıt Boyutları.

toplanan kayıtlarda şüpheli aktiviteleri açığa çıkarabilir. Bu nedenle, bu araçların yüklü olduğu ve yüklü olmadığı senaryoların karşılaştırılması, kayıt verilerinin stabil seviyesini ve sistem davranışlarının nasıl etkilendiğinin anlaşılmasında önemli fırsatlar sunar. Farklı senaryolarda elde edilen kayıt verilerinin analizi, normal ve zararlı dosyaların davranışlarının çeşitliliğini ortaya koyarak, geliştirilecek çözümlerin çeşitli ortamlarda daha uyumlu çalışmasına olanak tanır. Sonuç olarak, bu çoklu senaryo yaklaşımı, sistemin genel güvenilirliğini artırmak ve makine öğrenmesi modellerinin farklı koşullara genelleştirilebilmesini sağlamak açısından kritik bir öneme sahiptir [13].





4. ÖZNİTELİK ÇIKARIMI VE MÜHENDİSLİĞİ

Deneysel ortamlar üzerinde yapılan çalışmalar ve farklı senaryolarda gerçekleştirilen testler sonucunda, 7.000'den fazla dosyadan elde edilen olay kayıtları bir araya getirilmiştir. Toplanan dosya kayıtları, her kanalın özel evtx dosyasında tutulacak şekilde yapılandırılmıştır. Bir dosyada birden fazla kayıt bulunabileceğinden, her evtx dosyasındaki tüm kayıtlar ilk adımda birleştirilip tek bir dosyada toplanmalıdır. Bu sayede, farklı kaynaklardan gelen kayıt verilerinin eksiksiz ve bütüncül bir şekilde analiz edilebilmesi sağlanır.

Her bir dosya için toplanan kayıtlar, farklı kanallara ayrılmış olup, özellikle zararlı ve zararsız dosyaların farklı davranış kalıplarını yansıtmaktadır. Artık odak noktası, bu kayıtların hangi formatta tutulduğu ve nasıl en iyi şekilde işlenebileceğidir. Elde edilen kayıtların birçoğu, Windows olay kayıtlarının JSON benzeri yapılara dönüştürülmüş hali veya Sysmon gibi araçlardan alınan çıktılarından oluşmaktadır. Aşağıda, Kayıt 1 bu tür bir kayıt örneğinin JSON formatına dönüştürülmüş haline yer verilmektedir.

Kayıt 1'de görüldüğü üzere, "Event" anahtarının altında hem "System" hem de "EventData" bölümleri yer almaktadır. "System" bölümü, olayın hangi kanal üzerinden geldiği, hangi makinede kaydedildiği ve zaman damgası gibi temel bilgileri içerirken, "EventData" bölümü, çalıştırılan dosyanın konumu (*Image*) ve oturum açan kullanıcının kimliği (*User*) gibi detayları barındırır. Bu veriler, dosyaların çalışma şekilleri, kullanıcı erişimleri ve ağ aktiviteleri gibi pek çok kritik bilgiyi ortaya çıkarır.

Ancak, tüm anahtar kelimeler yalnızca bu örneklerle sınırlı kalmamaktadır. Örneğin, mevcut veriler esas alınarak belirlenen anahtar kelimeler mevcut olsa da, farklı analiz ortamlarında yeni anahtar kelimeler ortaya çıkabilir ve sisteme uyum sağlanmalıdır. Bu nedenle, toplanan tüm veriler detaylı şekilde incelenmeli ve olası tüm anahtar kelimeler tespit edilerek veri kümesine dahil edilmelidir. Böylelikle, analiz sürecinde elde edilen veriler, farklı senaryolara esnek bir şekilde uyarlanabilir ve sistemin genel performansı artırılabilir. Ayrıca sayı bazlı olarak incelenirse öznitelikler ve zararlı, zararsız arasında

örüntüler bulunabilir. Bu da hangi özniteliklerin önemli olduğunu destekler [50].

Kayıt 1: Örnek JSON Kaydı.

```
1 {
2   "Event": {
3     "@xmlns": "http://schemas.microsoft.com/win/2004/08/events/
      event",
4     "System": {
5       "Provider": {
6         "@Name": "Microsoft-Windows-Sysmon",
7         "@Guid": "5770385f-c22a-43e0-bf4c-06f6589ffbd9"
8       },
9       "EventID": "5",
10      "TimeCreated": {
11        "@SystemTime": "2024-09-06T07:08:31.2454672Z"
12      },
13      "Computer": "WIN10"
14    },
15    "EventData": {
16      "Data": [
17        {
18          "@Name": "Image",
19          "#text": "C:\\Windows\\System32\\taskhostw.exe"
20        },
21        {
22          "@Name": "User",
23          "#text": "NT AUTHORITY\\SYSTEM"
24        }
25      ]
26    }
27  }
28 }
```

4.1 Sayısal Bazlı Öznitelik Analizi

Çizelge 4.1 içinde, her öznitelik için zararlı ve zararsız dosya sayıları, kayıt sayıları yer almaktadır. Bu veriler, sistemdeki olay kayıtlarının farklı alanlardaki dağılımını ve yoğunluğunu ortaya koyarak, hem zararlı hem de zararsız yazılım verilerinin kalitesini ve çeşitliliğini değerlendirmemize olanak tanır.

Her anahtar kelime, kayıt dosyalarındaki dizin yapısına göre adlandırılmıştır. Örneğin, "Event/System/Provider" dizini, "Event_System_Provider" olarak isimlendirilmiştir. Tek bir dizin gibi değerlendirilerek, analiz sürecinde karşılaştırmalı çalışmalar yapıl-

Çizelge 4.1: Kayıt Verilerinden Çıkarılan Özniteliklerin Dosya ve Kayıt Sayıları.

Öznitelik	Zararlı Dosya Sayısı	Zararsız Dosya Sayısı	Zararlı Kayıt Sayısı	Zararsız Kayıt Sayısı
Event_System_Provider_@Name	2772	4732	2835485	963676
Event_System_Provider_@Guid	2772	4732	2816762	925254
Event_System_EventID	2767	4732	2741712	834511
Event_System_Version	2772	4732	2835485	963676
Event_System_Level	2772	4732	2835485	963676
Event_System_Task	2772	4732	2835485	963676
Event_System_Opcode	2772	4732	2835485	963676
Event_System_Keywords	2772	4732	2835485	963676
Event_System_TimeCreated_@SystemTime	2772	4732	2835485	963676
Event_System_EventRecordID	2772	4732	2835485	963676
Event_System_Correlation	2770	4732	2666230	674303
Event_System_Execution_@ProcessID	2772	4732	2835485	963676
Event_System_Execution_@ThreadID	2772	4732	2835485	963676
Event_System_Channel	2772	4732	2835485	963676
Event_System_Computer	2772	4732	2835485	963676

masına olanak tanır.

Verilere bakıldığında, bazı anahtar kelimelerin neredeyse her dosyada yer aldığı gözlemlenmektedir. Eğer bu kelimeler, içerikleri tamamen aynı veya çok benzer ise, ileride kullanılacak makine öğrenmesi modeli için değersizleşir, fakat tüm dosyalarda farklılık yaratmaması, davranışsal varyasyon sağlamaz. Zararlı ve zararsız yazılımların dosya sayıları göz önünde bulundurulduğunda, her tür dosyada bu anahtar kelimenin bulunma olasılıkları önemli bir gösterge haline gelir. Aynı zamanda, kayıt sayılarına göre hesaplanan oranlar da, bu kelimelerin veri setinde ne kadar etkili olduğunu ortaya koyabilir.

Bu tür kullanımların incelenmesine **sayısal bazlı analiz** denir. Sayısal bazlı analizde, belirli bir anahtar kelimenin kayıt dosyalarında ne sıklıkla yer aldığı, bu sıklığın hedeflenen sonuçlarla olan ilişkisi detaylı olarak değerlendirilir. Böylece, hangi anahtar kelimelerin modelin performansına katkıda bulunduğu, hangilerinin ise veri setinde homojenlik yarattığı belirlenebilir. Bu analiz yöntemi, veri ön işleme ve model optimizasyon aşamalarında, en etkili özniteliklerin seçilmesi için kritik bir rol oynar [51].

Eşsiz anahtar içeriğine bakmak, her bir anahtar kelimedenden ne kadar farklı değer elde edildiğini gösterir ve bu da modelde kullanılabilecek potansiyel belirtici faktörlerden biri olarak değerlendirilebilir. Sonuç olarak, düşük sıklığa sahip özniteliklerin seçimi, doğru eşik değerleriyle desteklendiğinde, modelin genel performansını artırarak, zararlı ve zararsız dosyaların davranışlarını daha net ayırt etmesine yardımcı olur.

Çizelge 4.2 içinde ilk olarak, "Event_System_TimeCreated_@SystemTime" özelliği neredeyse full eşsiz değer oranına sahiptir. Ancak, bu durum aslında beklenen bir so-

Çizelge 4.2: Kayıt Verilerinden Çıkarılan En Yüksek Eşsiz Anahtar Kelime Zararlı Dosya Oranları.

Öznitelik	Zararlı Yazılımlarda Eşsiz Anahtar Oranı
Event_System_TimeCreated_@SystemTime	0.9964059071354964
UtcTime	0.4275207142620348
ProcessGuid	0.4068498310342565
Event_System_EventRecordID	0.0098261814729468
ProcessId	0.0096911076266996
Value	0.0091762047141125
ParentProcessGuid	0.0070065590166906
CommandLine	0.0068365705208913
Message	0.0032855037901792
Event_UserData_CompatibilityFixEvent_StartTime	0.0022627514295609
Event_System_Correlation_@ActivityID	0.0016621489226186
metadata_format	0.0013941172694910
ParentCommandLine	0.0013927065765798
Image	0.0013655507380393
Hashes	0.0013084176751358

nuçtur çünkü bu öznitelik yalnızca olayların oluşturulma zamanını belirtir. Her bir kayıt, farklı bir zaman damgası içerdiğinden, bu yüksek eşsiz değer oranı, aslında belirli bir davranışa işaret etmez. Sistemin çalıştığı zamanı kaydetmek normal ve öngörülebilir bir durumdur.

Diğer oranlara bakıldığında, bazı özniteliklerin kayıt verilerindeki benzersiz değer sayıları düşük kalmaktadır ve bu öznitelikler tek başına modelde güçlü bir ayırım yapıcı özellik taşımamaktadır. Ancak, öznitelik kümesi içerisinde tamamlayıcı bir etken olarak değerlendirilirlirse kullanılabilirler. Özellikle, zararlı dosyalar için belirli bir eşik oranı tanımlanarak, bu eşik değerin altında kalan anahtar kelimeler seçilebilir. Böylece model, gereksiz ve tekrarlayan bilgileri ezberlemek yerine daha anlamlı varyasyonları öğrenebilir. Bununla birlikte, veri kümesine aşırı odaklanarak yapılan özellik mühendisliği, çözümün ezberlemeye dayalı çalışmasına neden olabilir. Bu yüzden, analizde rakamlara bakarken genelleme yapabilmek kritik öneme sahiptir.

Çizelge 4.3 içinde yer alan veriler, belirli anahtar kelimelerin zararlı ve zararsız yazılımlar üzerindeki benzersiz değer sayılarını göstermektedir. Örneğin, "ScriptBlockText" ve "ScriptBlockId" anahtar kelimeleri yalnızca zararlı yazılımlarda bulunmuş olup, zararsız yazılımlarda hiç kayıt edilmemiştir. Bu durum, zararlı yazılımların bu tür kayıtlarda belirgin izler bıraktığını gösterir ve bu anahtar kelimelerin, zararlı dosyaların tespitinde güçlü bir ayırt edici öznitelik olarak kullanılabileceğini ortaya koyar. Ayrıca,

Çizelge 4.3: Zararlı ve Zararsız Yazılımlarda Bazı Anahtar Kelimelerin Benzersiz Değer Sayısı.

Anahtar	Eşsiz Zararlı Yazılım Anahtar Değeri Sayısı	Eşsiz Zararsız Yazılım Anahtar Değeri Sayısı
ProcessGuid	1153617	100246
UtcTime	1212229	129268
Event_System_TimeCreated_@_SystemTime	2825295	948620
ProcessId	27479	1059
Event_System_EventRecordID	27862	2301
ScriptBlockText	455	0
ScriptBlockId	432	0
CurrentDirectory	215	14
param1	166	2
Event_UserData_LogFileCleared_Channel	164	0

"Event_UserData_LogFileCleared_Channel" anahtarının da zararsız kayıtlarda yer alması, bu özelliğin agresif programların davranışlarıyla yakından ilişkili olduğunun önemli bir ipucudur.

Diğer yandan, "ProcessGuid", "UtcTime" gibi anahtar kelimeler hem zararlı hem de zararsız dosyalarda yüksek eşsiz değer oranlarına sahiptir. Bu, özneteliklerin her iki durumda da sıkça ve çeşitli değerler içerdiğini göstermektedir. Dolayısıyla, bu tür anahtarlar tek başına ayırıcı güç taşımayabilir. Ancak, bazı durumlarda zararlı dosyalarda eşsiz değer sayısının daha yüksek olması, bu özelliğin zararlı davranışları yansıtmaya potansiyelini artırabilir. Öte yandan, "ProcessId" ve "Event_System_EventRecordID" gibi anahtarların benzersiz oranları oldukça düşük olup, bu durum, ilgili alanların genellikle sabit veya az varyasyon gösterdiğini, dolayısıyla modelde önceliklendirme yaparken bu anahtarların dikkatli değerlendirilmesi gerektiğini ortaya koymaktadır.

Genel olarak, bu tablonun incelenmesi, özellik mühendisliği sürecinde hangi anahtar kelimelerin daha değerli bilgiler sağladığını ve hangilerinin model için ayırıcı bir güç taşımadığını belirlemek açısından kritik önem taşır. Özellikle, bazı anahtar kelimeler zararlı dosyalarda bulunup zararsız dosyalarda bulunmaması, modelin eğitimi sırasında bu tür özelliklere daha yüksek ağırlık verilmesi gerektiğini gösterir. Buna karşılık, bazı anahtarlar düşük sayıda benzersiz değere sahip olduğundan, tüm dosyalarda

düzenli olarak görülmeyebilir ve bu durum, modelin ezberleme eğiliminde kalmadan genelleme yapabilmesi için, kullanılacak özniteliklerin seçimi sırasında dikkatle ele alınmalıdır. Böylece, farklı dosya ve kayıt sayılarına dayalı olarak, en etkili ve ayırt edici öznitelikler belirlenerek, modelin performansı optimize edilebilir.

4.2 Bağlamsal Analiz

Bağlamsal analiz kapsamında, zararlı yazılımların hangi dizinleri hedef aldığı önemli ipuçları sunmaktadır [52]. Zararlı yazılımlar, çeşitli dizin kategorilerini kullanarak dosyaları saklar veya komutlarını çalıştırır. Bu dizinler, sistemdeki erişim izinleri, izlenme sıklığı ve kullanılabilirlik gibi özellikleri nedeniyle saldırganlar için ilgi çekicidir. Aşağıda, yazılımın sıklıkla hedef aldığı dizin kategorileri ve nedenleri açıklanmıştır:

- **Geçici ve Önbellek Dizinleri:** \AppData\Local\Temp\ gibi geçici klasörler, zararlı dosyaları saklamak veya komut dosyalarını çalıştırmak için tercih ettiği dizinlerdir. Bu dizinler, çok sayıda süreç tarafından yazılabilir olduğundan ve genellikle daha az denetlendiğinden, saldırganlar için cazip hale gelir.
- **Sistem ve Program Veri Dizinleri:** C:\ProgramData\ ve C:\Program Files\Common Files\ gibi dizinler, meşru yazılımların da kullanımı nedeniyle yaygın olarak bulunur. Yazılım bu dizinleri kalıcılık sağlamak amacıyla kullanır. Çünkü bu dizinler, geniş erişim imkanları sunar ve normal sistem işleyişine entegre olduklarından, tespit edilmesi zorlaşır.
- **Yönetimsel ve Yüksek Ayrıcalıklı Dizinler:** C:\Users\Administrator\ gibi dizinler, yüksek ayrıcalık gerektiren ve yönetimsel erişim hakkı bulunan alanlardır. Yazılım bu dizinleri kullanarak sistemde daha geniş kontrol elde etmeye çalışır ve kullanıcı seviyesindeki kısıtlamaları aşar.
- **Alışılmadık ve Rastgele Dizinler:** Salırganlar, tespit edilmemek amacıyla standart olmayan dizinler oluşturabilirler. Örneğin, C:\tfsjike\, C:\mshyperbrowserDhcp\ veya C:\gijbzzhxajxjk\ gibi dizinler, meşru sistemlerde bulunmayan ve genellikle zararlı yazılım tarafından sabit olarak kodlanmış dizinlerdir.
- **Sistem Kurtarma ve Önyükleme Dizinleri:** C:\Recovery\WindowsRE\ veya C:\Boot\ gibi dizinler, sistem kurtarma süreçlerini veya önyükleme dizinlerini içermektedir. Zararlı yazılım, bu dizinleri manipüle ederek sistem bütünlüğüne

zarar verebilir veya önyükleme sürecini etkileyebilir [53].

- **Gizli veya Sistem Bileşenleri Dizinleri:** C:\Blockprovider\ veya C:\runtimebrokermonitor\ gibi dizinler, genellikle sistemle ilişkili isimler altında gizlenmiş olup, program için saklanabilecek gizli alanlar olarak kullanılmaktadır.

Bu dizinlerin analiz edilmesi, programın hangi dizinleri tercih ettiğini ve bu dizinlerde ne tür dosyaların saklandığını ortaya koyarak, normal sistem davranışlarından farklılıklarını belirlemede önemli rol oynamaktadır. Bağlamsal analiz, bu tür dizinlerin içerdiği bilgilerin incelenmesiyle, programın gizli aktivitelerini ve kaçınma stratejilerini daha net şekilde ortaya çıkarmaktadır. Özellikle, sistem ve program veri dizinleri gibi yaygın dizinlerin yanı sıra, saldırganların oluşturduğu alışılmadık dizinlerin varlığı, yazılımın tespit edilmesinde kritik ipuçları sunar. Bu nedenle, elde edilen veri kümesinde, bu dizinlerden alınan veriler, diğer özniteliklerle birlikte değerlendirilerek, makine öğrenmesi modellerinin zararlı ve zararsız dosyalar arasındaki ayrımı daha doğru yapabilmesine olanak tanır.

4.3 Örüntü Analizi

Bir diğer önemli teknik, örüntü analizidir. Bu yöntem, bir sürecin gerçekleştirdiği olayların ya da eylemlerin sırasını inceleyerek, normal davranıştan sapmaları tespit etmeye odaklanır. Örneğin, zararsız ve zararlı dosyalarda belirli komutların frekansı benzer olsa bile, bu komutların çalıştırılma sırası veya aralarındaki zaman farkları incelendiğinde, zararlı dosyalara özgü gizli davranışlar ortaya çıkabilir. Böylece, yalnızca komutların sayısına bakmak yerine, komutların birbirleriyle olan sıralı ilişkisi de analiz edilerek, zararlı yazılımın gizleme stratejileri daha net şekilde gözlemlenebilir [54].

4.4 Anormal Analizi

Anormal tespit yöntemleri de dinamik analiz sürecinde önemli bir yer tutar. Normal sistem davranışını modelleyerek, veri seti içerisindeki alışılmadık ya da nadir görülen davranışlar tespit edilebilir. Bu sayede, zararsız ve zararlı dosyalar arasında benzer sıklıkta gözükse dahi, normal davranıştan sapmalar yakalanabilir. Örneğin, bir dosyada belirli bir komut dizisinin sıradışı bir şekilde tekrarlanması veya beklenenden farklı

zaman aralıklarıyla çalışması, o dosyanın zararlı olabileceğine işaret edebilir [55].

4.5 Söz Dizimi, Entropi ve Karmaşıklık Analizi

Söz konusu teknikler, komut satırı parametreleri, işlem isimleri ve dosya yolları gibi söz dizimi (syntax) tabanlı özniteliklerin de analiz edilmesiyle desteklenir. Söz dizimi tabanlı öznitelikler, belirli komutların ve argümanların yapısını inceleyerek, zararlı yazılımın genellikle kullandığı karmaşıklıklaştırma (obfuscation) stratejilerini tespit edebilir. Örneğin, benzer komutların zararsız yazılım örneklerinde sabit bir sırayla yürütülürken, zararlı setlerde bu sıra bozuluyorsa, bu durum zararlı niyetin bir göstergesi olarak değerlendirilebilir. Son olarak, entropi ve karmaşıklık tabanlı analiz, veri setindeki belirsizlik ve rastgelelik ölçümlerine dayanır. Entropi, verinin ne kadar rastgele veya öngörülemez olduğunu hesaplayan bir parametredir. Yüksek entropi değeri, veride daha fazla rastgelelik olduğunu, dolayısıyla zararlı yazılımın sıklıkla karmaşıklıklaştırma (obfuscation) teknikleri kullanarak veriyi karıştırdığını gösterebilir. Bu parametre sayesinde, zararsız ve zararlı örnekler arasındaki ince farklar tespit edilerek, modelin yazılım davranışlarını doğru şekilde ayırt etmesi sağlanır. Bu tekniklerin bir arada kullanılması, sadece belirli komutların frekansına değil, aynı zamanda bu komutların sıralı ilişkisine, anomali tespitine ve entropi ölçümlerine dayalı olarak yazılım davranışlarının daha kapsamlı ve güvenilir bir şekilde analiz edilmesini mümkün kılmaktadır. Böylece, elde edilen öznitelik seti, hem zararsız dosyaların hem de zararlı dosyaların davranışsal farklılıklarını en iyi şekilde yansıtarak, modelin genel doğruluğunu artırmaya katkıda bulunur [56].

4.6 Öznitelik Seçimi

4.6.1 Sayısal Bazlı Öznitelikler

- **Süreç Ortalamaları:** "ParentProcessId" ve "ProcessId" alanları kullanılarak hesaplanır. Her üst sürecin oluşturduğu alt süreç sayısının ortalamasını belirleyerek, sistemdeki süreç oluşturma yoğunluğunu ve programın agresif süreç yaratma eğilimini ortaya koyar.
- **Hata Kodu Oranı:** Olay kayıtlarında karşılaşılan "ErrorCode" alanının eşsiz sayısının toplam kayıt içindeki oranını ifade eder. Bu oran, sistem kararsızlığı ya

da potansiyel saldırı girişimlerine işaret edebilir, dolayısıyla sistemdeki olağan dışı durumların tespitinde yardımcı olur.

- **"Log File Cleared Channel" Oranı:** "UserData → LogFileCleared → Channel" alanı üzerinden, kayıt dosyasının kaç kez temizlendiğini ölçer. Yüksek temizlik sayıları, programın izlerini gizleme amacıyla kayıtların silinmesine yönelik stratejiler uyguladığının göstergesidir.
- **Eşsiz Erişilen Dosya Dizin Oranı:** "Image" alanı kullanılarak, sürecin dosya sistemi ile etkileşiminde kaç farklı dosya yoluna eriştiği belirlenir. Bu öznetelik, sistemdeki dosya erişim çeşitliliğini ve genişliğini ortaya koyar.
- **"Compatibility Fix Event FixID" Oranı:** "UserData → CompatibilityFixEvent → FixID" alanı kullanılarak, uyum düzeltme olaylarının sayısı izlenir. Bu sayı, yazılım kaçınma stratejileri kapsamında sistemin uyumluluk sorunlarına müdahale etme eğilimini gösterebilir.
- **"Interface Guid" Oranı:** "InterfaceGuid" alanından elde edilen benzersiz ağ arayüzlerinin oranını hesaplar. Sistem tarafından kullanılan farklı ağ arayüzlerinin çeşitliliğini yansıtarak, ağ etkileşimlerinin kapsamını belirler.
- **Hash Oranı:** "Hashes" alanını temel alır. Dosya hash değerlerinin sayısal dağılımını ölçerek, süreç oluşturma anormallikleri veya dosya modifikasyonlarını tespit etmede önemli bir gösterge sunar.
- **"Script Block" Oranı:** "ScriptBlockText" alanı kullanılarak, PowerShell script bloklarının kayıtlardaki oranı hesaplanır. Özellikle dosya içi karmaşıklık (obfuscation) tekniklerinin belirlenmesinde kritik bir rol oynar.

4.6.2 Bağlamsal Öznetelikler

- **Şüpheli Süreç Dizinleri:** "ProcessPath" ve "CurrentDirectory" alanları kullanılarak, süreçlerin çalıştırıldığı dizinlerin şüpheli olup olmadığı değerlendirilir. Zararlı yazılımlarda, genellikle normal sistem yollarından saparak, güvenlik zafiyetine sahip dizinlerde çalıştırılabilir dosyalar üretebilir. Bu öznetelik, şüpheli dizinlerin varlığını ortaya koyarak, zararlı faaliyetlerin tespitinde önemli bir gösterge sunar.
- **Süreç İsimleri Kullanımları:** "Image" alanına dayalı olarak, sistemde sıkça kar-

şılaşılan, kötü niyetli uygulamalarda yaygın olan süreç isimlerinin sayısı hesaplanır. Zararlı yazılımlar, bazen meşru süreç isimlerini kullanarak kendini gizlemeye çalışsa da, zararsız dosyalarda bu isimlerin tutarlı kalması, ayırt edici bir öznitelik olarak değerlendirilebilir.

- **"Original File" Anahtarları:** "OriginalFileName" alanı, dosya adlarında bulunan "install", "update" gibi şüpheli kelimelerin varlığını tespit eder. Bu tür anahtar kelimeler, dosyanın potansiyel zararlı aktivitelerle ilişkili olup olmadığını belirlemek açısından önemli ipuçları sunar.
- **"Privilege Event" Oranı:** "EventId" alanındaki 4672 ve 5379 gibi yetki yükseltme olaylarının oranı hesaplanır. 4672, yeni bir oturum açıldığında özel ayrıcalıkların atanmasını, 5379 ise uygulamanın Credential Manager üzerinden kimlik bilgilerini okuma işlemini ifade eder. Bu oran, potansiyel yetki suiistimali veya saldırı girişimlerinin tespiti için kritik bir göstergedir.
- **Hash Anormalleri:** "Hashes" ve "Image" alanlarından elde edilen veriler kullanılarak, dosya hash değerlerindeki olağan dışı sapmalar tespit edilir. Bu anormallikler, dosya bütünlüğünde bozulma veya yazılım tarafından gerçekleştirilen modifikasyonların işareti olarak değerlendirilebilir.
- **Terminal Şüpheli Anahtar Sayı Oranı:** "CommandLine", "ProcessCommandLine" ve "ParentCommandLine" alanları üzerinden, şüpheli argümanlara sahip benzersiz komut satırı girişlerinin sayısı belirlenir. Çizelge 4.4 içerisinde görülen komutlar bunlara örnektir.

Çizelge 4.4: Zararlı Yazılımlar Tarafından Kullanılan Şüpheli Komut Argümanları.

schtasks
msedge.exe
powershell
cmd.exe
vbs
wscript
cscript
ps1 /c
/b /s -EncodedCommand
-nop -w hidden %TEMP%
AppData Startup
rundll32.exe

Bu anahtar kelimeler ve hash değişiklikleri bağlamsal olarak potansiyel öznitelik çı-

karmada önemli özelliğe sahiptir.

4.6.3 Zaman Tabanlı Öznitelikler

Bu kategoride, kayıt verilerindeki olayların zaman damgalarından elde edilen bilgilerin analizi yapılır.

- **Ortalama Olay Zaman Aralığı:** Olay kayıtları arasındaki ortalama zaman farkını ölçer. Olay zaman damgaları, "System → TimeCreated → @SystemTime" alanından elde edilir. Bu ölçüt, sistem aktivitelerinin ani patlamalar veya düzenli bir akışla gerçekleşip gerçekleşmediğini ortaya koyar.
- **Olay Zaman Aralığı Standart Sapması:** Olaylar arasındaki zaman farklarının standart sapmasını hesaplar. Yüksek standart sapma, olay zamanlamalarında büyük varyasyon olduğunu, düşük değer ise olayların daha düzenli gerçekleştiğini gösterir.

4.6.4 İstatistiksel Öznitelikler

Belli anahtar kelimelerin entropileri hesaplanarak yeni öznitelikler oluşturulur.

- **İşlem Kimliği Entropisi:** İşlem kimliklerinin rastgeleliğini ölçer. Yüksek entropi, süreçlerin oluşturulma ve sonlandırılma kalıplarında büyük bir rastgelelik olduğunu gösterir. Zararlı yazılımların agresif süreç yaratma eğiliminde olabileceğine işaret edebilir.
- **Olay Sayılarının Ortalaması:** Olay kayıtlarındaki olay sayılarını medyan değer üzerinden değerlendirir. Bu yöntem, uç değerlerin etkisini azaltarak, sistemdeki genel olay yoğunluğunu belirlemede daha sağlam bir gösterge sunar.
- **Opcode Entropisi:** İşlemlerde kullanılan opcode'ların çeşitliliğini ölçer. Yüksek değer, yazılımın dosya içi karmaşıklıklaştırma (obfuscation) teknikleri uyguladığını gösterebilir.
- **"Rule Names" Entropi Oranı:** İşlemlerde kullanılan kural isimlerinin entropisini hesaplar. Bu oran, sistemdeki yapılandırmalarda anormallik olup olmadığını ve zararlı yazılımın olağan dışı davranışlar sergileyip sergilemediğini değerlendirir.

dirmeye yardımcı olur.

Bunlar dışında birçok anahtar kelimenin entropisinde öznitelik setine eklenmiştir.

4.6.5 Söz Dizimi Tabanlı Öznitelikler

Kelimelerin içerisindeki detaylara göre öznitelikler üretilmiştir.

- **Terminal Komutu Bayrak(Flag) Sayısı:** Komut satırında kullanılan bayrak sayısını hesaplar. Bu sayı, komut yürütme sürecindeki karmaşıklığı ortaya koyar.
- **Terminal Komutu "Obfuscation" Bayrağı:** Yazılımın kendi faaliyetlerini gizlemek için kullandığı karmaşıklıklaştırmaya (obfuscation) bayraklarının oranını belirler. Yüksek oran, saldırganın komutları gizlemek için yoğun şekilde karmaşıklıklaştırmaya (obfuscation) tekniklerine başvurduğunu gösterir.
- **Terminal Komutu Rakam Sayısı:** Komut satırında kullanılan rakamların oranını ölçer. Bu oran, özellikle otomatik üretilen veya karmaşık komutların varlığını ortaya koyarak, zararlı davranışların tespitine katkıda bulunur.
- **Terminal Komutu Büyük Harf Sayısı:** Komut satırındaki büyük harflerin sayısını hesaplar. Büyük harf kullanımı, imza tabanlı tespit mekanizmalarını aşmak amacıyla gerçekleştirilebilecek manipülasyonları gösterebilir.
- **Terminal Komutu Özel Karakter Sayısı:** Komut satırında yer alan özel karakterlerin sayısını ölçer. Özel karakter kullanımı, terminal komutu karmaşıklıklaştırmaya (obfuscation) ve yazılım kaçınma stratejilerinin bir göstergesi olabilir.

4.6.6 Örüntü Bazlı Öznitelikler

- **"Process Burstiness" Oranı:** Süreç oluşturma olaylarının ani ve yoğun patlamalarını ölçer. Bu oran, her iki dosya türünde de gözlemlenen süreç oluşturma zaman farklarına dayanır. Zararlı yazılım, saldırı niyetiyle ani süreç artışları sergileyebilir, bu da normal sistem davranışlarından belirgin şekilde ayrışır.

4.7 Öznitelik Değerlendirme

Veri kümesinin üzerinden elde edilen bazı öznitelikler yukarıda detaylı olarak açıklanmıştır. Diğer öznitelikler ise benzer kalıplara sahip olmakla birlikte, farklı anahtar kelimelerden türetilmiştir. Bu özniteliklerin potansiyel bir makine öğrenmesi modeli girdisi olarak kullanılmadan önce, öznitelik setinin genel performansının doğrulanması ve optimize edilmesi gerekmektedir. Özniteliklerin geçerliliğini değerlendirmek için ilk adım, başlangıçtaki öznitelik setinde yer alan verilerin incelenmesidir. Bu aşamada, modelin performansını olumsuz yönde etkileyebilecek, gereksiz veya aşırı korelasyona sahip özellikler ayıklanarak, daha anlamlı bir öznitelik kümesi elde edilmeye çalışılır. Verideki her bir anahtar kelimenin sıklığı ve varyansı, zararsız ve zararlı dosyalar arasındaki farkların ne kadar belirgin olduğunu ortaya koyar. Öznitelik geçerliliğinin ölçülmesinde yaygın olarak kullanılan yöntemlerden biri korelasyon analizidir [57]. Bu analiz, her bir öznitelik çifti arasındaki ilişkiyi ölçerek, özniteliklerin birbirleriyle ne kadar yüksek oranda korelasyon gösterdiğini ortaya koyar. Eğer iki öznitelik arasında yüksek bir korelasyon varsa, bu durum "multikolinerlik" olarak adlandırılır ve modelin karar verme sürecinde bu özniteliklere düşük önem verilebilir. Çünkü benzer bilgiyi ifade eden iki öznitelik arasından ayırım yapmak zorlaşır, bu da modelin genel öngörü gücünü azaltır.

Bu sorunu aşmak için, korelasyon katsayısı belirli bir eşik değerin üzerinde olan öznitelikler tespit edilir. Genellikle, yüksek korelasyona sahip iki öznitelik arasında, hedef sınıfa daha güçlü bir ilişki sunan öznitelik tercih edilir. Diğer öznitelik kaldırılır veya ikisinin birleşimiyle tek bir öznitelik oluşturulur. Ancak, öznitelik setinin yalnızca korelasyon analizine dayanılarak optimize edilmesi, veri setinin yerel örüntülerine aşırı uyum sağlayan (overfitting) bir model oluşturma riskini beraberinde getirebilir. Yüksek korelasyon, bazı durumlarda, farklı senaryolarda hâlâ değerli bilgiler sunabilir.

Bu nedenle, öznitelik seçiminde korelasyon analizi, çapraz doğrulama ve test veri setleriyle test gibi doğrulama tekniklerinin birlikte kullanılması gerekmektedir. Bu bütüncül yaklaşım, modelin çeşitli senaryolarda genelleme yapabilme yeteneğini artırarak, seçilen özniteliklerin hem lokal örüntülerden ziyade genel geçerliliğe sahip olmasını sağlar. Özellikle, benzer anahtar kelimelerin tüm dosyalarda aynı değeri üretmesi, ayırım yapıcı gücün düşük olmasına neden olduğundan, modelin performansını iyileştirir.

mek için bu tür özniteliklerin dikkatlice seçilmesi önemlidir.

4.8 Zayıf Öznitelikleri Eleme

Özellik mühendisliği sürecinde, korelasyonu yüksek olan özniteliklerin işlenmesinin yanı sıra, hedef değişkenle yani yazılımın zararlı veya zararsız olmasıyla zayıf ilişki gösteren özniteliklerin de elenmesi önemli bir adımdır. Bu zayıf öznitelikler, kayıt verilerinde çok düşük varyasyon veya neredeyse sabit değerler sunabilir. bu durumda, modelin karar verme sürecinde anlamlı bir katkı sağlamazlar. Ancak, sadece mevcut veri setine dayalı olarak bu özniteliklerin elenmesi, ilerleyen aşamalarda farklı veri setlerinde daha değerli bilgiler sunabilecek potansiyel özelliklerin göz ardı edilmesine yol açabilir. Bir öznitelik, yerel bir veri setinde zayıf görünse dahi, farklı senaryolarda veya test veri setleriyle test edildiğinde daha anlamlı hale gelebilir. Bu nedenle, öznitelik seçimi sırasında dikkatli bir doğrulama süreci uygulanmalıdır. Özellikle, bir öznitelik yalnızca sınırlı bir örüntüye dayanıyorsa, genel genelleme kabiliyetini artırmak adına, çapraz doğrulama ve test veri setleriyle test edilerek, gerçekten önemli olup olmadığı değerlendirilmelidir.

Çizelge 4.5, zararlı yazılım sınıfına ait kayıt verilerinde yer alan en yüksek korelasyona sahip öznitelikleri göstermektedir. Örneğin, "System_Provider_Entropy", "Opcode_Entropy" ve "HC_State_Ratio" gibi özniteliklerin korelasyon oranları sırasıyla 0.299, 0.270 ve 0.260 gibi yüksek değerler sunmaktadır. 7.000'den fazla örnek içeren bu veri setinde, 0.2'nin üzerinde korelasyon oranları, bu özniteliklerin zararlı yazılım davranışlarını ayırt etmede güçlü adaylar olduğunu göstermektedir.

Çizelge 4.5: Öznitelik Korelasyon Değerleri.

Öznitelik	Korelasyon Değeri
System_Provider_Entropy	0.299009
Opcode_Entropy	0.270205
HC_State_Ratio	0.260328
Avg_Metadata_Char	0.251334
Message_Count	0.233152
Process_ID_Entropy	0.219696
Line_Numbers	0.215942
Keyword_Entropy	0.206790
Avg_Metadata_Digit	0.181909
Cmd_Num_Digits	0.177724

Özellikle dikkat çeken nokta, en yüksek korelasyon değerlerinin büyük oranda bağlamsal ve söz dizimi (syntax) tabanlı analiz özniteliklerinden elde edilmiş olmasıdır. Bu öznitelikler, zararlı ve zararsız dosyalar arasında belirgin farklılıklar ortaya koyarak, modelin ayırım yapıcı gücünü artırmaktadır. Örneğin, "Process_ID_Entropy" ve "Opcode_Entropy" gibi öznitelikler, süreçlerin rastgeleliğini ve dosya içi komut çeşitliliğini ölçerken, zararsız dosyalarda bu değerler çok daha düşük seyredebilir. Bu durum, zararlı yazılımın karakteristik davranış kalıplarını net bir şekilde ortaya koymaktadır.

Ancak, zayıf özniteliklerin elenmesi sürecinde, sadece korelasyon analizine dayanmadan, özniteliklerin bağlamsal anlamı ve genel veri setine katkısı da dikkate alınmalıdır. Özellikle, bazı öznitelikler mevcut veri setinde düşük korelasyon gösterse bile, farklı ortamlarda daha anlamlı varyasyonlar sunabilir. Bu nedenle, öznitelik seçiminde, yalnızca yerel örüntülere dayalı olarak karar verilmemelidir. Aynı zamanda, geniş veri setleri ve farklı senaryolarda doğrulama yapılarak, hangi özniteliklerin modelin genel performansını artırdığı belirlenmelidir.

- **"System Provider" Entropi (0.299009):** Bu öznitelik, kayıt verilerinde yer alan Event_System_Provider_@Name anahtar kelimesinden türetilmiştir. Sağlayıcı isimlerindeki entropiyi (rastgeleliği) hesaplayarak, yüksek entropinin yazılım isimlerini rastgeleleştirerek veya manipüle ederek kendini gizleme çabalarına işaret edebileceğini gösterir.
- **"HC State" Oranı (0.260328):** Bu oran, olay kayıtlarında kullanılan HCState anahtar kelimesi üzerinden elde edilmiştir. Sistem kayıtlarında, durum (state) tanımlayıcılarının kullanımını izler. Yüksek bir oran, yazılım sistemde yüksek ayrıcalıklı durumları kullanarak zararlı aktiviteler gerçekleştirdiğine dair ipuçları sunar.
- **"Message" Oranı (0.233152):** Bu öznitelik, Message anahtar kelimesinden türetilmiştir. Sistem olayları sırasında üretilen toplam mesaj sayısını ifade eder. Yüksek mesaj sayısı, sistemde anormal aktivitenin ve yazılımın aşırı etkinlik yaratma girişimlerinin göstergesi olabilir.
- **"Process ID" Entropi (0.219696):** Bu ölçüt, ProcessID anahtar kelimesi kullanılarak hesaplanır. İşlem kimliklerinin rastgeleliğini ölçer. Yüksek entropi, sü-

reçlerin düzensiz ve öngörülemez bir biçimde oluşturulduğunu, bu durumun ise yazılım tarafından sıklıkla gözlemlenen bir davranış kalıbını yansıttığını gösterir.

- **"Line Numbers" (0.215942):** Bu öznelik, olay kayıtlarındaki LineNumbers anahtar kelimesinden elde edilir. Benzersiz satır numaralarının sayısını hesaplayarak, zararlı yazılımın, tehlikeli olmayan dosyalara süreçlere kıyasla daha çeşitli veya beklenmedik satır numaraları üretme eğilimini ortaya koyar. Bu durum, davranış farklılıklarının tespitinde yararlı bir gösterge sunar.
- **"Keyword" Entropi (0.206790):** Bu öznelik, Keyword anahtar kelimesi üzerinden hesaplanır. Olay kayıtlarında yer alan belirli anahtar kelimelerin entropisini ölçer. Yüksek bir değer, bu anahtar kelimelerde rastgelelik olduğunu ve zararlı yazılımın, bu kelimeleri gizlemek amacıyla karmaşıklıklaştırmaya (obfuscation) stratejilerine başvurduğunu gösterebilir.

4.9 Özneliklerin Birbiriyle Olan Korelasyonlarının İncelenmesi

Çizelge 4.6, zararlı yazılım sınıfına ait kayıt verilerinde yer alan öznelikler arasındaki yüksek korelasyon değerlerini göstermektedir. Bu yüksek korelasyon, bazı özneliklerin benzer veya örtüşen bilgileri yakaladığını ve dolayısıyla büyük ölçüde birbirinin yerine kullanılabileceğini işaret eder. Örneğin, "File_Version_Diff" ile

"File_Version_Build_Info" arasındaki korelasyon değeri 0.994797 olup, bu iki öznelik dosya sürüm bilgilerini farklı açılardan ifade etmektedir. Biri sürüm farklarını, diğeri ise ek yapı bilgilerini yansıtmaktadır. Bu durum, benzer versiyonlama kalıplarının tespitinde her iki özelliğin de birbirinin yerini alabileceğini göstermektedir. Aynı şekilde, "Service_Entropy" ile "Service_Proportion" arasındaki yüksek korelasyon değeri (0.993845), çalıştırılan dosyaların sistemde kullandığı hizmet isimlerinin çeşitliliğini ve oranını ölçmektedir. Bu iki öznelik, hizmetlerin varyabilitesini belirlerken, tehlikeli programların saldırı stratejilerinde benzer yapısal değişiklikleri yansıtmaları nedeniyle oldukça yüksek bir örtüşme göstermektedir. Benzer şekilde, "Subject_Event_Ratio" ile "Privilege_Event_Ratio" arasındaki korelasyon (0.986755) da, yetkiyle ilgili olayların izlenmesinde, hem kullanıcı olaylarının hem de yetki yükseltme olaylarının birbirine yakın sonuçlar vermesi nedeniyle dikkat çekicidir.

Bunun yanı sıra, "Line_Numbers_Count" ile "Message_Count" arasındaki korelasyon değeri (0.967701) da önemli bir göstergedir. Her iki öznelik de, süreç çalıştırmaları

Çizelge 4.6: Yüksek Korelasyona Sahip Öznitelik Çiftleri.

Öznitelik 1	Öznitelik 2	Korelasyon Değeri
File_Version_Diff	File_Version_Build_Info	0.994797
Service_Entropy	Service_Proportion	0.993845
Subject_Event_Ratio	Privilege_Event_Ratio	0.986755
Product_Entropy	Company_Entropy	0.974575
Line_Numbers	Message_Count	0.967701
Event_Kurtosis	Event_Kurtosis_Skewness	0.961278
Cmd_Flag_Extension	Desc_Entropy	0.957698
Avg_Metadata_Digit	Avg_Metadata_Char	0.956810
Service_Entropy	Avg_Permission_Flag_Count	0.952827
Unique_Process_Guid	Utc_Ratio	0.952732
Desc_Entropy	Product_Entropy	0.952020

sırasında üretilen olay kayıtlarının satır sayısını ölçmekte ve zararlı yazılımın yoğun sistem mesajları üretme eğilimini yansıtmaktadır. Bu yüksek korelasyon, bu tür özniteliklerin ayrı ayrı modelde kullanılmasının, gereksiz bilgi tekrarı ve modelin karmaşıklığına yol açabileceğini göstermektedir. Öznitelik çıkarım aşamasında, bu yüksek korelasyon değerlerine sahip öznitelikler, özellikle geleneksel makine öğrenmesi algoritmalarında modelin verimliliğini azaltabilecek potansiyel sorunlar doğurabilir. Derin öğrenme modelleri genellikle bu tür fazlalıkları tolere edebilse de, veri setinde yer alan yüksek korelasyon, modelin aşırı ezber yapmasına ve yeni veriler üzerinde genelleme yapamamasına yol açabilir. Bu nedenle, öznitelik seçiminde, yüksek korelasyona sahip özniteliklerin, modelin hedef değişkenle olan ilişkisine göre değerlendirilip, gereksiz olanların kaldırılması veya birleştirilmesi gerekmektedir. İlk olarak, derin öğrenme dışı algoritmaların performansını ve genelleme kabiliyetini artırmak amacıyla, öznitelik iyileştirme süreci uygulandı. Bu süreç, modelin verimliliğini yükseltmek ve aşırı öğrenme riskini azaltmak için, başlangıçta elde edilen öznitelik setindeki gereksiz tekrar eden bilgilerin minimize edilmesini hedeflemektedir. Aynı zamanda, yazılım türü tespiti için kritik olan temel bilgileri koruyarak, modelin daha ayrımcı hale gelmesi sağlanmaya çalışılmıştır.

4.10 Yüksek Korelasyon İlişkisi İçeren Öznitelikleri Düzeltme

Bu iyileştirme süreci birkaç aşamadan oluşmaktadır. İlk adımda, korelasyon analizi kullanılarak, birbirleriyle 0.9'un üzerinde yüksek korelasyon gösteren öznitelikler be-

lirlenmiştir. Bu aşama, hangi özniteliklerin benzer veya örtüşen bilgileri taşıdığını tespit etmekte ve potansiyel gerek olmayan alanları ortaya koymaktadır. Böylece, yüksek korelasyona sahip öznitelik çiftleri gözden geçirilmiş, modelin verimliliğini azaltabilecek fazla bilgilerin elimine edilmesi için zemin hazırlanmıştır [58].

4.11 Tekrar Eden Özniteliklerin Kaldırılması veya Birleştirilmesi

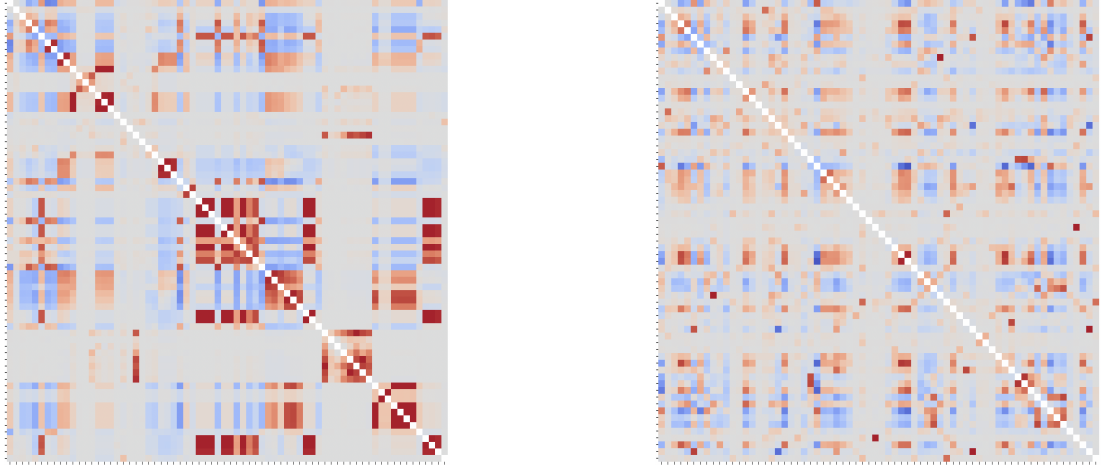
İkinci aşamada, benzer bilgileri yansıtan özellikler ya birleştirilmiş ya da gerekli olmayanlardan biri kaldırılmıştır. Bu işlem, önemli detayların kaybolmaması adına titizlikle yürütülmüştür. Bir öznitelik seti içerisinde, benzer bilgi sunan iki ya da daha fazla özellik bulunması durumunda, bu özniteliklerden en yüksek ayırım yapıcı güce sahip olanı tercih edilmiştir. Böylece, modelin karar verme sürecinde gereksiz bilgi tekrarı engellenmiş, ve her bir öznitelik daha anlamlı bir hale getirilmiştir.

4.12 Basit Modeller ile Test Etme

Üçüncü aşamada, iyileştirilmiş öznitelik seti, temel bir makine öğrenmesi modeli kullanılarak doğrulanmıştır. Bu doğrulama, modelin hem eğitim hem de test verileri üzerinde performansını artırırken, aynı zamanda aşırı öğrenme riskinin de minimize edilip edilmediğini ölçmüştür. İyileştirilmiş öznitelik seti, orijinal öznitelik setine alternatif olarak, daha verimli ve genelleyici bir model elde edilmesine katkı sağlamıştır.

Öznitelik iyileştirme sürecinde, yalnızca yerel veri setine dayanarak karar verilmemesi gerektiği de önemli bir husustur. Aşırı iyileştirme, modelin sadece eğitim verisine çok uyum sağlamasına ve farklı veri setleri üzerinde başarısız olmasına neden olabilir. Bu nedenle, öznitelik seçimi, farklı senaryolarda ve test veri setlerinde test edilerek, genelleme yeteneği yüksek bir model oluşturulması hedeflenmiştir.

Hem iyileştirilmiş hem de orijinal öznitelik setleri kullanılarak model performansının karşılaştırılması, hangi özniteliklerin model için daha güçlü ve ayırım yapıcı olduğunu belirlemekte kritik bir rol oynamıştır. Bu kapsamlı testler sayesinde, elde edilen öznitelik seti, zararsız ve zararlı dosyalar arasındaki ince farkların doğru şekilde yakalanmasına olanak tanıyan, esnek ve genelleyici bir modelin oluşturulmasına katkı sağlamıştır.



Şekil 4.1: Öznitelikler Korelasyon Matris Değişimi.

Şekil 4.1’de korelasyon matrislerinin sol tarafı, öznitelik iyileştirme işlemi öncesinde elde edilen korelasyon dağılımını göstermektedir. Bu matriste, birçok özelliğin birbirleriyle yüksek düzeyde korelasyon, koyu kırmızı veya koyu mavi alanlar, gösterdiği dikkat çekmektedir. Bu durum, modelde çoklu korelasyon sorununu doğurarak, aynı bilgiyi tekrar eden veya gereksiz yere karmaşıklığa sebep olan özelliklerin varlığına işaret eder. Ayrıca, aşırı korelasyona sahip öznitelikler, makine öğrenmesi algoritmalarında aşırı öğrenme (overfitting) riskini artırabilir ve modelin genelleme yeteneğini zayıflatabilir [59].

Sağ tarafta ise, öznitelik iyileştirme sonrasında elde edilen korelasyon matrisi yer almaktadır. Bu matriste, yüksek korelasyona sahip alanların önemli ölçüde azaldığı, renk yoğunluğunun daha seyrek ve daha açık tonlarda kaldığı görülmektedir. Böylece, benzer bilgiyi yansıtan veya birbirlerini tekrar eden özellikler ya tamamen kaldırılmış ya da birleştirilmiştir. Bu yaklaşım, hem modelin karar verme sürecinde gereksiz bilgi yükünü azaltmakta hem de her bir özelliğin ayırım gücünü daha net şekilde ortaya çıkarmaktadır. Dolayısıyla, sağdaki matriste gözlemlenen daha az yoğun korelasyon, öznitelik setinin daha tutarlı ve genelleyici bir yapıya kavuştuğunu göstermektedir.

Özetle, soldan sağa geçişteki fark, modelin yüksek oranda ilişki gösteren özniteliklerden arındırılmış veya bu özniteliklerin birleştirilmiş olması sayesinde, korelasyon dağılımının daha dengeli hale geldiğini kanıtlamaktadır. Bu durum, makine öğrenmesi modelinin farklı veri kümelerinde de iyi performans gösterme olasılığını, genelleme yeteneğini artırırken, aşırı öğrenme riskini düşürebilir.



5. MODEL EĞİTİMİ VE DEĞERLENDİRME

Bu bölümde, Windows olay kayıtlarına dayalı olarak zararlı yazılım sınıflandırması yapmak amacıyla üç farklı öznitelik seti üzerinde çeşitli makine öğrenmesi modelleri eğitilmiş ve değerlendirilmiştir. Söz konusu öznitelik setleri *Ham*, *Normal Düzey İyileştirme* ve *Agresif İyileştirme* olarak adlandırılmıştır. Bu yaklaşımın temel hedefi, her bir öznitelik setinin farklı model türleriyle, geleneksel makine öğrenmesi sınıflandırıcıları, sinir ağı tabanlı modeller ile yaklaşımlarında elde ettiği performansı karşılaştırarak, zararlı yazılım tespitinde hangi yöntemin daha başarılı olduğunu vurgulamaktır.

Değerlendirme sürecinde, tüm modeller için sağlam ve güvenilir bir performans ölçümü sağlamak amacıyla K-Katlamalı Çapraz Doğrulama (*K-Fold Cross-Validation*) yöntemi uygulanmış ve çoklu yinlemelerin ortalaması alınarak sonuçlar yorumlanmıştır. Üç öznitelik seti şu şekilde tanımlanmıştır:

- **İyileştirme İçermeyen Set:** Bu seçenekte, ham öznitelik setine herhangi bir değişiklik veya filtreleme uygulanmamıştır. Dolayısıyla, modelin eğitimi için temel bir karşılaştırma noktası sunar.
- **Normal Seviye İyileştirmiş Set:** Bu öznitelik setinde, 0.9'un üzerinde yüksek korelasyon gösteren öznitelikler tespit edilerek, gereksiz tekrarları önlemek amacıyla birleştirilmiş veya elenmiştir. Ancak, zararlı yazılım tespiti için kritik bilgilerin korunmasına özen gösterilmiştir.
- **Agresif İyileştirilmiş Set:** Daha agresif bir yaklaşım izlenerek, özniteliklerin birleştirilmesi veya kaldırılması işlemi daha kapsamlı yapılmıştır. Buradaki öncelik, aşırı öğrenmeyi (*overfitting*) engellemek ve basit bir öznitelik seti oluşturmaktır. Bu nedenle, yüksek korelasyonun yanı sıra, modelde düşük ayırım gücüne sahip öznitelikler de büyük ölçüde elenmiştir.

Çizelge 5.1 içerisinde görüldüğü gibi küme üzerinde yapılan işlemlerde beklenen rakamlara ulaşılmıştır. Agresifleştikçe sıfır oranı artıp, korelasyon düşmüştür. Bu üç farklı yaklaşım sayesinde, hangi iyileştirme düzeyinin zararlı yazılım için oluşturulan

sınıflandırmasında daha yüksek doğruluk ve genelleme performansı sağladığı incelenmiş, böylece ilerleyen bölümlerde detaylı olarak ele alınacak olan sonuçların temeli atılmıştır.

Çizelge 5.1: Veri kümelerine ait özet istatistikler.

Küme	Öznitelik	Sıfır Oran	Seyrek Oran	>0,9 Korr. Oran	Ort. Korr.	Aykırı Oran
Agresif	69	0,360	0,333	0,0017	0,131	0,0105
Normal	76	0,330	0,289	0,0049	0,171	0,0116
Ham	92	0,295	0,261	0,0103	0,193	0,0126

5.1 K-Fold Çapraz Doğrulama

Model performansını daha güvenilir şekilde değerlendirmek amacıyla, K-Katlamalı Çapraz Doğrulama (K-Fold Cross-Validation) yöntemi kullanılmıştır. Bu teknik, modelin yalnızca veri kümesinin belirli bir alt kümesi üzerinde iyi performans göstermesini değil, aynı zamanda farklı alt bölümler üzerinde genelleme yapabilmesini sağlar. Veri kümesi, eşit büyüklükte K parçaya bölünür ve model, her yinelemede bir parçayı test seti, kalan K-1 parçayı ise eğitim seti olarak kullanarak K defa eğitilir. Tüm iterasyonlardan elde edilen sonuçların ortalaması, modelin genel performansını daha doğru ve dengeli bir şekilde ölçmeyi mümkün kılar [60]. Örneğin, 5-Katlamalı Çapraz Doğrulama durumunda, veri kümesi beş eşit parçaya ayrılır; model her seferinde farklı bir parçayı test seti olarak kullanarak toplamda beş kez eğitilir. Bu yaklaşım, veri kümesindeki her örneğin hem eğitim hem de test aşamalarında değerlendirilmesini sağlayarak, modelin veri setinin tüm bölümlerinde tutarlı ve genelleme performansı göstermesini temin eder. Böylece, model yalnızca belirli bir alt küme üzerinde ezber yapmaz, genel veri dağılımı üzerinde sağlam öngörüler üretebilir.

5.2 Kullanılan Makine Öğrenmesi Modelleri Ön Bilgi

5.2.1 Random Forest

Random Forest modeli, birden fazla karar ağacının oluşturulması ve bu ağaçların çıktılarının ortalaması alınarak nihai kararın verilmesi prensibine dayanır. Model, tıpkı birden fazla kişiden alınan tavsiyelerin çoğunluğa göre karar verilmesi gibi, farklı ağaçlardan elde edilen sonuçların çoğunluk oyuna göre sınıflandırmayı yapar. Bu yöntem,

özellikle karmaşık veri setlerinde yüksek performans ve dayanıklılık sağlar [61].

5.2.2 Logistic Regression

Lojistik regresyon, her bir sınıfa ait olasılıkları hesaplayarak, en yüksek olasılığa sahip sınıfı etiket olarak atar. Bu yöntem, veri setindeki örneklerin belirli bir sınıfa ait olma olasılıklarını modelleyerek, doğrusal sınırlar çizebilir. Özellikle, sınıflar arasındaki ayrımın nispeten net olduğu durumlarda etkili sonuçlar verir [62].

5.2.3 Support Vector Classifier (SVC)

Destek Vektör Sınıflandırıcısı (SVC), iki sınıfı ayıran en iyi doğrusal sınırı belirlemeye çalışır. İki boyutlu uzayda, zararlı ve zararsız örnekleri birbirinden ayıran en uygun sınır çizgisini oluşturarak, veriler arasındaki ayrımı maksimize eder. Bu yöntem, özellikle yüksek boyutlu veri setlerinde, ayırt edici sınırların belirlenmesinde etkili sonuçlar sunar [63].

5.2.4 K-Nearest Neighbors (KNN)

K-En Yakın Komşu (KNN) modeli, bir örneği sınıflandırırken, o örneğe en yakın komşuların sınıf dağılımını göz önüne alır. Eğer komşuların çoğunluğu zararlı ise, yeni örnek zararlı olarak sınıflandırılır. Benzer şekilde, zararsız örneklerin yoğun olduğu bölgelerde yeni örnek zararsız olarak değerlendirilir. Bu yöntem, basit yapısı ve yorumlanabilirliği sayesinde, veri setindeki lokal benzerlikleri ortaya koymada başarılıdır.

5.2.5 Gradient Boosting

Gradient Boosting algoritması, birden fazla küçük karar ağacının ardışık olarak oluşturulması ve her bir ağacın hatalarını düzelterek nihai modeli güçlendirmesi prensibine dayanır. Her adımda, önceki adımın hatalarından yola çıkarak yeni bir model eğitilir. Böylece, modelin tahminleri zamanla iyileştirilir. Bu yöntem, özellikle karmaşık veri

setlerinde yüksek doğruluk oranları elde etmekte etkilidir [64].

5.2.6 RobustScaler

Veri setinde uç değerlerin etkisini azaltmak için kullanılan RobustScaler, verileri, medyan ve interkuartil aralık gibi merkezi eğilim ölçütlerine odaklanarak ölçeklendirir. Bu sayede, aşırı yüksek veya düşük değerlere sahip örneklerin model performansını olumsuz etkilemesi engellenir ve daha stabil bir veri seti elde edilir [65].

5.2.7 MinMaxScaler

MinMaxScaler, verileri genellikle 0 ile 1 arasındaki sabit bir aralığa indirger [65]. Bu yöntem, verilerin boyutunu küçültürken modellerin daha hızlı öğrenmesini sağlar. Ancak, veri setinde çok yüksek veya çok düşük değerlere sahip örnekler varsa, bu durum MinMaxScaler'ın duyarlılığını artırabilir. Dolayısıyla, modelin performansı bu tür uç değerlerden etkilenebilir.

5.2.8 Sinir Ağları

Bu bölümde, geleneksel makine öğrenmesi ve derin öğrenmenin basitleştirilmiş bölümünde kullanılan iyileştirilmiş ve ham öznitelik setleri üzerinde sinir ağları test edilmiştir. Sinir ağları, insan beyninin çalışma prensiplerinden esinlenerek geliştirilmiş modellere dayanır. Bu ağlar, birbirine bağlı nöronlar aracılığıyla verilerdeki desenleri öğrenir. Sinir ağları, çeşitli özelliklerin bulunduğu karmaşık veri setlerinde, doğrusal olmayan ilişkileri ve soyut desenleri öğrenebilme esnekliği sayesinde özellikle etkili sonuçlar verir. Bu nedenle, zararlı yazılım tespiti gibi, öznitelikler arasındaki ilişkilerin sezgisel olarak ortaya konulmasının zor olduğu durumlarda sinir ağları başarılı bir performans sergilemektedir.

5.2.9 MLP Sınıflandırıcısı

Çok Katmanlı Algılayıcı (MLP) sinir ağı, katman katman "açgözlü" bir yaklaşımla verilerden öğrenir [66]. Her katman, kendisine aktarılan bilgiyi işler ve bir sonraki katmana aktararak modelin orijinal verideki desenleri sürekli iyileştirmesine olanak tanır. MLP sınıflandırıcısı, hem iyileştirilmiş hem de ham veri setlerini kullanarak, doğrusal olmayan ve ince ilişkileri yakalamada etkili olmuş, zararlı ile zararsız dosyalar arasındaki ayrımında başarılı sonuçlar vermiştir.

5.2.10 Değerlendirme Metrikleri

Hem iyileştirilmiş hem de iyileştirilme uygulanmamış öznitelik setleri üzerinde K-Katlamalı Çapraz Doğrulama uygulandıktan sonra, tüm yinelemelerin sonuçları ortalama alınarak model performansı değerlendirildi. Bu yöntem, veri setinin belirli bir alt kümesinden kaynaklanabilecek anormalliklerin etkisini azaltmak amacıyla tercih edilmiştir. Aşağıda, her bir modelin performansını ölçmek için kullanılan metrikler yer almaktadır:

- **Ortalama Eğitim Doğruluğu (Average Train Accuracy):** Modelin eğitim verileri üzerinde elde ettiği doğruluk oranlarının tüm katlamalar(fold) boyunca ortalaması.
- **Ortalama Validasyon Seti Doğruluğu (Average Validation Set Accuracy):** Modelin doğrulama verileri üzerinde elde ettiği doğruluk oranlarının tüm katlamalar boyunca ortalaması.
- **Ortalama ROC AUC (Average ROC AUC):** Zararlı ve zararsız dosyaları ayırt etme yeteneğini ölçen ROC AUC değerlerinin ortalaması [67].
- **Ortalama Aşırı Öğrenme (Average Overfitting):** Eğitim ve test doğruluğu arasındaki farkın ortalaması, yüksek değerler modelin aşırı öğrenmeye eğilimli olduğunu gösterir.
- **Tam Aşırı Öğrenme (Full Overfitting):** Modelin aşırı öğrenme durumunu daha

kapsamlı ölçmek amacıyla aşağıdaki formül 5.1 ile hesaplanır:

$$\begin{aligned} \text{Tam Aşırı Öğrenme} = & \left(\text{Eğitim Doğruluğu} - \text{Test Doğruluğu} \right) + \\ & \left(\text{Eğitim Doğruluğu} - \text{Doğrulama Seti Doğruluğu} \right) + \\ & \left(\text{Eğitim Doğruluğu} - \text{En Düşük Test Sonucu Doğruluğu} \right) \quad (5.1) \end{aligned}$$

- **Test Doğruluğu (Test Accuracy):** Eğitim ve doğrulama aşamalarında kullanılmayan, tamamen bağımsız bir veri seti üzerinde ölçülen doğruluk oranıdır. Bu metrik, modelin

- Farklı koşullarda toplanmış verilerle,
- Kontrollü ortamdan sapmalar içeren deneysel düzeneklerle
- Gerçek dünya senaryolarına daha yakın durumlarla

ne kadar iyi genelleme yapabildiğini gösterir. Örneğin, test verileri:

- İnternet bağlantısı kapalı bir sanal makineden,
- Sandbox ortamı her seferinde yeniden başlatılmayıp, ardışık çalıştırmalarla
- Farklı sistem kaynaklandırma ayarlarıyla
- Süreçler düzenlenmeden(Sandbox kaçınma teknikleri kullanılmadan) önceki haliyle

toplanmış olabilir. Bu tür sürpriz değişkenler, test doğruluğunun yalnızca modelin "öğrendiği" verilere değil, aynı zamanda beklenmeyen veya kontrol dışı koşullara karşı da sağlam olduğunu kanıtlamasını sağlar. Böylece, yüksek test doğruluğu; modelin ortamın "stabil" olmadığı durumlarda bile tutarlı performans gösterme yeteneğinin güçlü bir göstergesi haline gelir.

- **Karmaşıklık Matrisi (Confusion Matrix):** İkili sınıflamada modelin tüm kararları dört kutuda özetlenir:

- **TP** – Yakalanan zararlı dosyalar
- **FP** – Yersiz alarm (*zararsız* dosyayı zararlı sanma)
- **FN** – Kaçırılan zararlı (en riskli hücre)
- **TN** – Doğru "temiz" karar

- **F1-Skoru:** Precision (isabet) ile Recall (yakalama) dengesini tek değerde birleştirir:

$$\text{Precision} = \frac{TP}{TP+FP}, \quad \text{Recall} = \frac{TP}{TP+FN}, \quad F1 = \frac{2 \text{Precision} \text{Recall}}{\text{Precision} + \text{Recall}}.$$

F1 yükselmesi için hem *yanlış alarmın* (FP) düşük hem de *zararlı kaçırmanın* (FN) az olması gerekir; bu yüzden güvenlik senaryolarında en dengeli tekil başarı ölçütü olarak kullanılır.

5.2.11 Aykırı Değerlerin (Outlier) Temizlenmesi

Veri setindeki sayısal özelliklerde yer alan uç değerlerin model performansını bozmasını engellemek amacıyla üç farklı outlier temizleme yöntemi karşılaştırılmıştır.

IQR Yöntemi: Her bir sayısal özellik için Q_1 (25. yüzde birlik çeyrek) ve Q_3 (75. yüzde birlik çeyrek) değerleri hesaplanarak

$$IQR = Q_3 - Q_1$$

ara değeri bulunur. Aykırı değer sınırları ise

$$\text{ALT SINIR} = Q_1 - k \times IQR, \quad \text{TAVAN SINIR} = Q_3 + k \times IQR$$

formülleriyle belirlenmiştir; burada $k = 1.5$ kullanılmıştır. Bu eşiklerin dışında kalan tüm satırlar

$$\text{mask}_{IQR} = \neg((x < \text{ALT SINIR}) \vee (x > \text{TAVAN SINIR}))$$

şeklinde filtrelenerek temiz veri elde edilmiştir.

Z-Skor Yöntemi: Özniteliklerin ortalama μ ve standart sapma σ değerleri üzerinden her bir gözlem için

$$z = \frac{x - \mu}{\sigma}$$

skoru hesaplanır. Geleneksel olarak $|z| > z_{\text{eik}}$ koşulunu sağlayan veriler aykırı kabul edilir; bu çalışmada $z_{\text{eik}} = 3$ alınmıştır. Filtreleme maskesi

$$\text{mask}_z = |z| \leq 3$$

olarak uygulanmış, bu dışındaki satırlar veri setinden çıkarılmıştır. **Isolation Forest Yöntemi:** Isolation Forest, rastgele alt örneklemeler üzerinde ağaç yapıları kurarak verileri "izole etme" mesafelerini ölçer. Aykırı değerler, normal gözlemlere göre daha kısa bir yol boyuyla (path length) ağacın yapraklarına ulaşır. Modelin önemli parametreleri şunlardır:

- `n_estimators`: Ağaç sayısı,
- `max_samples`: Her ağaç için kullanılan örneklem büyüklüğü,
- `contamination`: Veri içinde beklenen aykırı oranı.

Her gözlem için *anomaly score* hesaplanır ve belirlenen `contamination` oranına göre

$$\text{mask}_{\text{IF}} = (\text{score} \leq \text{threshold})$$

şeklinde temizlenmiş veri seçilmiştir. Bu üç yöntemin çıktıları ayrı ayrı uygulanıp, eğitim ve doğrulama aşamalarındaki performans metrikleri (Eğitim doğruluğu, Doğrulama doğruluğu, ROC AUC, Overfit ölçütü, En düşük katman doğruluğu ve Harici test doğruluğu) karşılaştırmalı olarak raporlanmıştır. Bu metrikler, modelin hem eğitim hem de test aşamalarındaki performansını kapsamlı bir şekilde değerlendirmemize olanak sağlar. Özellikle, aşırı öğrenme durumunu tespit etmek için kullanılan metrikler sayesinde, modelin yalnızca eğitim verilerine çok uyum sağlayıp yeni verilere genelleme yapamaması engellenir. Ayrıca, test doğruluğu gibi metrikler, modelin gerçek dünya koşullarında ne kadar başarılı olduğunu ölçmede önemli bir rol oynar. Bu yaklaşımla, hem derin öğrenme hem de geleneksel makine öğrenmesi algoritmaları için elde edilen sonuçlar, karşılaştırmalı olarak değerlendirilebilir ve en uygun öznelilik seti ile model optimizasyonu sağlanabilir.

5.3 Sonuçların Yorumlanması

Tablolarda sunulan sonuçlar, farklı aykırı değer temizleme ile öznelilik iyileştirme stratejileri (Agresif İyileştirme, Normal İyileştirme, Ham Öznelilik Seti) ve kullanılan ölçekleyici yöntemleri (Min-Max Ölçekleyici, Robust Ölçekleyici) kullanılarak eğitilen sınıflandırıcıların performansına ilişkin önemli bilgiler sunmaktadır. Değerlendirme, eğitim doğruluğu, validasyon seti doğruluğu, ROC AUC, ortalama aşırı öğrenme, tam aşırı öğrenme ve test doğruluğu gibi temel metrikler üzerinden yapılmış. Sonuçların

tüm K-Katlamalı Çapraz Doğrulama iterasyonlarının ortalaması alınarak sunulması, modelin veri seti üzerindeki genelleme yeteneğini sağlamlaştırmıştır.

5.3.1 Aykırı Değer Temizleme Değerlendirme

Tablo 5.2'deki sonuçlardan aykırı temizleme metotlarının veri seti üzerindeki etkilerinin sonuçları çıkarılır.

- Isolation Forest her kümede yalnızca %10 satırı siler, IQR ise %65–70 aralığında veri kaybına yol açar.
- Özellik ortalamalarındaki en küçük kayma Normal + Isolation Forest'ta 0.116; en büyük kayma Agresif + IQR'de 0.422'dir.
- Sınıf dengesindeki değişim: Isolation Forest zararlı oranını yaklaşık 1 puan oynatır; Normal + IQR temizliği bu oranı 9.68 puan azaltır.
- Veri kaybı sıralaması Agresif > Ham > Normal'dir; nedeni Agresif'teki polinomik ve birleşmiş özniteliklerin yüksek ve alçak değerleri ve Ham'daki yüksek öznitelik sayısıdır.
- Uygulama önerisi: üretim senaryolarında Isolation Forest, orta seviye temizlikte Z-Score, yalnızca prototip aşamasında IQR kullanılmalıdır.

Geri kalan kısımda istatistikler IQR dışındaki aykırı değer temizleme metotları ile devam edilecek. IQR'ın veri kaybının yüksekliği modellerde kullanılmaması kararına yöneltti.

5.3.2 Model Sonuçları

Tablo 5.3'deki sonuçlardan model metotlarının veri seti üzerindeki etkilerinin sonuçları çıkarılır.

- **RandomForest** – Aşırı öğrenme en düşük (0,0416); doğrulama 0,857 ve test 0,963 ile istikrarlı. Bagging tabanlı ağaçlar uzun kuyruklu özniteliklerde ezber yapmadığı için veri kaybından az etkilenir. İlk testlerde yüksek overfit oranları ile karşılaşıldı fakat ağaç derinliği azaltıldıkça overfit ve doğrulama oranları daha dengelendi.

Çizelge 5.2: Aykırı değer temizleme kurallarının her küme üzerindeki etkisi.

Küme	Kural	Silinen Satır	% Veri Kaybı	Medyan $ \Delta\mu /\mu$	Zararlı % Azalma Oranı
Agresif	IQR	5193	69,53	0,422	7,23
	Z-Score ($ z >3$)	1478	19,79	0,271	2,98
	Isolation Forest (10 %)	747	10,00	0,163	1,42
Ham	IQR	5007	67,04	0,329	9,06
	Z-Score ($ z >3$)	1433	19,19	0,204	3,57
	Isolation Forest (10 %)	747	10,00	0,158	1,29
Normal	IQR	4917	65,83	0,284	9,68
	Z-Score ($ z >3$)	1426	19,09	0,181	3,59
	Isolation Forest (10 %)	747	10,00	0,116	1,04

Çizelge 5.3: Sınıflandırıcıların ortalama (*ort*) ve standart sapma (*ss*) değerleri – aşırı öğrenme artan sırada.

Sınıflandırıcı	Doğrulama ort (ss)	Aşırı Öğrenme ort (ss)	Test ort (ss)
RandomForest	0.8572 (0.0013)	0.0416 (0.0200)	0.9627 (0.0048)
XGBoost	0.9112 (0.0028)	0.0550 (0.0367)	0.9736 (0.0097)
SVC	0.7728 (0.0086)	0.0616 (0.0031)	0.8195 (0.0094)
MLPClassifier	0.9166 (0.0237)	0.0640 (0.0300)	0.9657 (0.0347)
GradientBoosting	0.8848 (0.0049)	0.1010 (0.0520)	0.9314 (0.0343)
LogisticRegression	0.8336 (0.0267)	0.1573 (0.0240)	0.9285 (0.0191)

- **XGBoost** – Doğrulama 0,911, test 0,974; aşırı öğrenme 0,055. Shrinkage ve L1/L2 cezaları, polinomlu (Agresif) sütunlardaki uç değerleri dengeleyerek en yüksek test başarımını verir. XGBoost test kümesinde en yüksek başarıyı sağlamıştır, çünkü:
 - Ağaç tabanlı yapısı ölçek farklılıklarına karşı duyarsızdır; özniteliklerde ek ön işleme gerektirmez.
 - Gradient boosting, her iterasyonda sadece artık hatayı hedeflediğinden nadir fakat bilgilendirici EventID sıçramalarını yakalar, ancak shrinkage katsayısı ezberi sınırlar.
 - L1/L2 gömülü ceza ve erken durdurma, 70–90 boyutlu özellik uzayında gereksiz dallanmaları budar; ham, polinom ve oran sütunlarının birleşiminden doğan gürültü genelleme kaybına dönüşmez.
- **SVC** – Doğrulama 0,773, test 0,820; overfit 0,062. Uzun kuyruklu yüksek boyutta RBF çekirdeği ölçek duyarlı kalır; bu nedenle genel başarı düşüktür, ancak az ezber yaptığı için overfit küçük görünür.

- **MLPClassifier** – Doğrulama 0,920, test 0,966; overfit 0,064. Gizli katmanlar polinomik ilişkileri yakalar. Daha az veri seti agresifliğinden etkilenir. Fakat standart sapma yüksek, katlamalar arası oynaklık var. MLPClassifier doğrulama kümesinde en yüksek skoru vermiştir, çünkü:
 - Ağın gizli katmanları, polinomik ve oranlı özelliklerde gömülü doğrusal olmayan ilişkileri doğrudan temsil edebilir; bu nedenle beş katlı iç doğrulamada hızlıca en iyi desenleri öğrenir.
 - Aşırı parametre kapasitesi sayesinde eğitim/validasyon dağılımı arasındaki küçük farkları bile yakalar; ancak aynı esneklik test kümesinde hafif aşırı öğrenmeye (0,06) yol açar.
- **GradientBoosting** – Doğrulama 0,885, test 0,931; overfit 0,101. Sınırlı shrinkage'ı ağaç sayısını arttırınca ezberi tetikliyor; XGBoost'a göre daha çok genelleme, testte geride olmaktadır.
- **LogisticRegression** – Doğrulama 0,834, test 0,93; overfit 0,16 (en yüksek). Lineer model, kuyruklu dağılımı tek ağırlıkla temsil etmeye çalışınca zararsız patlama özniteliklerde büyük katsayılar yükler; testte düşüş sınırlı görünse de saha verisinde risklidir.

5.3.3 Veri Küme Sonuçları

Tablo 5.4'deki sonuçlardan veri kümelerinin ortalama başarı oranları çıkarılır.

- **Agresif küme**
 - Doğrulama ortalaması 0,8833 ile *en yüksek* değeri verir; sapma 0,0387 makul düzeydedir.
 - Aşırı öğrenme ortalaması 0,0711 ile kümeler arasındaki en düşük değerdir; karmaşık polinom/oran özellikleri ezbersiz kullanılmıştır.
 - Test ortalaması 0,9561 ile de açık ara en iyi performansa sahiptir; ek özelliklerin genelleme gücü yüksektir.
- **Ham küme**
 - Doğrulama ortalaması 0,8800; sapma 0,0574 ile en yüksek varyansa sahiptir.
 - Test ortalaması 0,9148; ham öznitelik bolluğu bazı katlamalarda faydalı,

genel başarımı ise sınırlı tutuyor.

- **Çıkarım**

- Karmaşık fakat iyi ölçeklenmiş özellikler (Agresif) yüksek doğrulama + düşük overfit kombinasyonu sağlayarak gerçek veride avantaj getiriyor.
- Ham sayımlar model varyansını artırıyor; bu nedenle en iyi sonuç için ham + istatistik + polinom karışımını dengeli tutmak gerekiyor.

Çizelge 5.4: Veri kümelerinin ortalama (*ort*) ve standart sapma (*ss*) değerleri – aşırı öğrenme artan sırada.

Veri Kümesi	Doğrulama ort (ss)	Aşırı Öğrenme ort (ss)	Test ort (ss)
Agresif	0.8833 (0.0387)	0.0711 (0.0424)	0.9561 (0.0380)
Ham	0.8800 (0.0574)	0.0956 (0.0658)	0.9148 (0.0510)

5.3.4 En İyi Model Çıkarımları

Tablo 5.5 ve 5.6'deki metriklere bakarak en iyi sonuç veren modellerden varsayımlar yapılabilir.

- **Agresif kümesinin ilk 6'da beş satırı kaplamasının nedeni**

Polinom ve oran sütunları, zararlı–zararsız ayrımını tek başlarına başlatabilecek "yerel" desenler üretir. Z-Score ile uç değerler kırılıp Robust ölçeklemeyle aralık daraltılınca, bu sütunlar ezber riski olmadan modele güçlü sinyal taşır. Böylece en kötü katlamada bile doğruluk %90'ın altına düşmez; Ham ve Normal kümeleri aynı barajda kalamaz. Genel olarak ham veri setinin örüntüsel ilişkilerde daha büyük potansiyele sahip olduğu fakat agresif veri setinin daha stabil ve daha fazla modelde başarılı olduğu görülmüştür. Bu sonuçların öznitelik sayısı farkı ve agresif setin daha güçlü ve güvenilir özniteliklere sahip olmasıyla alakalı olduğu düşünülebilir.

- **XGBoost'un dört konfigürasyonla listede yer alması**

Gradient-boosted ağaçlar, veride "tek sütunluk karar kuralları"nı art arda toplayarak artık hatayı küçültür. Shrinkage + alt-örnekleme + L1/L2 cezası sayesinde polinom patlamalarını dal sayısını kısırarak dengeler; overfit 0.03–0.10 bandında kalır. Model ölçeklerden az etkilenir, bu yüzden MinMax ile Robust arasında test

Çizelge 5.5: En İyi Model performans metrikleri — *En Düşük Başarı* azalan sırada.

Metot	Küme	Ölçekleyici	Sınıflandırıcı	Eğitim	Doğrulama	ROC AUC	Overfit	En Düşük	Test
isoforest	Ham	Robust	MLPClassifier	0,9981	0,9608	0,9880	0,1115	0,9520	0,9635
Z-Score	Agresif	Robust	MLPClassifier	0,9772	0,9374	0,9775	0,0222	0,9224	0,9868
Z-Score	Agresif	MinMax	XGBoost	0,9781	0,9149	0,9706	0,1072	0,9082	0,9588
Z-Score	Agresif	Robust	XGBoost	0,9774	0,9142	0,9713	0,0970	0,9057	0,9639
isoforest	Agresif	MinMax	XGBoost	0,9653	0,9114	0,9681	0,0334	0,9053	0,9780
isoforest	Agresif	Robust	XGBoost	0,9646	0,9095	0,9679	0,0290	0,9026	0,9790

Çizelge 5.6: Seçili konfigürasyonların ortalama karışıklık matrisi bileşenleri ve başlıca performans metrikleri.

Metot	Küme	Ölçekleyici	Sınıflandırıcı	TN	FP	FN	TP	İsabet(Zararsız)	İsabet(Zararlı)	F1
isoforest	Ham	Robust	MLP	4560	127	163	2544	0.966	0.953	0.946
Z-Score	Agresif	Robust	MLP	3823	142	233	1793	0.943	0.927	0.905
Z-Score	Agresif	MinMax	XGBoost	3819	146	364	1662	0.913	0.920	0.867
Z-Score	Agresif	Robust	XGBoost	3815	150	364	1662	0.913	0.918	0.866
isoforest	Agresif	MinMax	XGBoost	4500	180	475	2239	0.905	0.926	0.872
isoforest	Agresif	Robust	XGBoost	4488	192	477	2237	0.904	0.921	0.870

skorları çok yakındır.

- **Agresif dışında listeye giren tek satır: Ham + IsoForest + Robust + MLP**

Ham kümede 92 öznitelik, MLP'nin daha deterministik ilişki araması sayesinde daha kullanışlı hale gelmektedir. IsoForest patlama veya tekrarlama yaşayan "EventID" bazlı özniteliklerin aşırı uç satırlarını temizler, ama frekans farkının kendisini korur. RobustScaler uç değer etkisini bastırır.. Aynı öznitelik dağılımı teste de sızdığı için yüksek overfit (0,11) genel başarıyı düşürmez ve Ham kümesinin tek üst-sıra sonucu ortaya çıkar.

Test Doğruluğunun Doğrulama Başarısından Yüksek Çıktığı Durumlar

Test setinde elde edilen doğruluk oranlarının doğrulama setini aşmasının nedeni, test verilerinin *önlem alınmayan, stabil olmayan ortamlarda* toplanmış olmasına dayanmaktadır. Bu durum, aşağıdaki şekilde özetlenebilir:

- **Patlama(Burst) Olay Yoğunluğu:** Test ortamındaki olay günlüklerinde, kısa

zaman aralıklarında art arda gerçekleşen ve "burst" olarak tanımlanan yüksek yoğunluklu olay kümeleri fazlasıyla mevcuttur.

- **Dinamik örüntülerin genellenmesi:** Model, eğitim ve doğrulama seti aşamalarında bu burst davranışlarını yalnızca gürültü olarak değil, *anlamlı sinyal* olarak öğrenmiştir.
- **Stabil olmayan ortamlardan kaynaklanan çeşitlilik:** Gerçek dünya test verisi, sandbox benzeri kontrollü ortamların aksine işlem sırası, dosya yolu ve zamanlama bakımından büyük sapmalar içerir. Model bu dinamik kalıpları iyi kavramış olduğu için, test ortamındaki dalgalı örüntülere karşı daha yüksek doğruluk sergiler.



6. SONUÇLAR VE TARTIŞMA

Bu çalışmada, Windows olay kayıtlarını kullanarak dinamik zararlı yazılım tespiti performansını artırmaya yönelik kapsamlı bir analiz gerçekleştirilmiştir. Çalışmanın temel amacı, gerçek zamanlı davranış analizinde sağlam ve ölçeklenebilir bir çözüm elde etmek için veri toplama, sandbox kurulumu, özellik mühendisliği ve model eğitimi sırasında alınan kararların etkilerini ortaya koymaktır. Aşağıda, gerçekleştirilen deneylerin, veri toplama ve otomasyon süreçlerindeki zorlukların, öznetelik çıkarımındaki stratejilerin ve model eğitimindeki yaklaşımların detaylı bir değerlendirmesi sunulmaktadır.

6.1 Sandbox Kurulumu ve Kaçınma Teknikleri

Çalışmanın temel unsurlarından biri, yazılımın güvenli bir şekilde çalıştırılabileceği kontrol altındaki izole bir sandbox ortamının oluşturulmasıdır. Sandbox, ağ ve cihaz güvenliğini tehlikeye atmadan zararlı yazılımların davranışlarını yakından gözlemleyebilmek amacıyla özenle yapılandırılmıştır. Bu ortamın oluşturulmasında, fiziksel makinelerle benzer donanım kimlikleri elde etmek için *donanım karmaşıklıklaştırma (obfuscation)* teknikleri uygulanmış. Böylece sanal makinenin gerçek bir fiziksel sistem gibi görünmesi sağlanmıştır. Ayrıca, Microsoft Office, Adobe Reader gibi yaygın kullanılan uygulamaların yanı sıra, Steam ve Epic Games gibi eğlence uygulamalarının kurulması, sistemin tipik kullanıcı ortamını simüle ederek yazılımın analiz ortamını tespit etme olasılığını azaltmıştır. Son olarak, Sysmon ve gelişmiş ağ kayıt araçları kullanılarak, yazılımın çalışma sırasında ortaya çıkardığı detaylı davranışsal desenler kaydedilmiştir.

6.2 Veri Toplama ve Otomasyon Zorlukları

Veri toplama aşamasında, zararlı ve zararsız olmak üzere toplam 7.000'den fazla dosya üzerinde çalışmak, büyük hacimli kayıt verilerinin yönetimi açısından önemli zorluklar ortaya çıkarmıştır. Bu yüksek veri hacmi nedeniyle, her dosyanın çalıştırılması sonrasında sandbox ortamının temiz bir duruma geri döndürülmesi (anlık görüntü geri

yüklemesi) kritik hale gelmiştir. Ancak, bazı yazılım örnekleri kayıt aktarımı tamamlanmadan sistemi yeniden başlatmış ya da dosya sistemini bozmuş, bu durum kayıt verilerinin eksik veya zarar görmesine yol açmıştır. Bu sorunun aşılabilmesi için, çoklu sanal makine yapılandırmaları kullanılarak kayıt verilerinin ayrı bir sunucu VM'ye otomatik olarak aktarılması sağlanmış, böylece veri bütünlüğü korunmuştur.

6.3 Öznitelik Çıkarım Süreci

Ham kayıt verilerinden anlamlı özniteliklerin çıkarılması süreci, hem bağlamsal hem de sayısal yaklaşımlar kullanılarak titizlikle gerçekleştirilmiştir. Bu aşamada, öznitelikler altı ana kategoriye ayrılmıştır: sayı bazlı, bağlamsal, zaman tabanlı, istatistiksel, söz dizimi tabanlı ve desen sırası öznitelikleri. Örneğin, sayı bazlı öznitelikler süreç oluşturma yoğunluğu, hata kodu oranları ve dosya yolu erişim çeşitliliği gibi ölçütler üzerinden temel sistem aktivitesini ortaya koyarken, bağlamsal öznitelikler, süreçlerin çalıştırıldığı dizinler, komut satırı argümanları ve dosya adlarında yer alan şüpheli kelimeler üzerinden yazılım davranışlarını belirlemeye yardımcı olmuştur.

Öznitelik iyileştirme sürecinde, tekrarlanabilir özniteliklerin tespit edilip birleştirilmesi veya elenmesi için korelasyon analizi uygulanmış, modelin aşırı öğrenme riski azaltılmış ve daha genelleşici bir öznitelik seti elde edilmiştir. Bu iyileştirme, hem eğitim hem de test veri setleri üzerinde model performansını optimize etmek açısından büyük önem taşımaktadır.

6.4 Model Eğitimi ve Değerlendirme

Model eğitimi aşamasında, ham ve iyileştirilmiş öznitelik setleri üzerinde çeşitli makine öğrenmesi ile derin öğrenme modelleri eğitilmiş; performansları K-Katlamalı Çapraz Doğrulama yöntemiyle değerlendirilmiştir. Eğitim doğruluğu, doğrulama doğruluğu, ROC AUC, ortalama aşırı öğrenme, tam aşırı öğrenme ve test doğruluğu gibi metrikler, modellerin hem mevcut veri kümesine hem de yeni veriler üzerindeki genelleme kabiliyetine ışık tutmuştur.

Sonuçlar, konfigürasyona göre farklı metriklerde anlamlı gelişmeler elde edildiğini göstermektedir. Aykırı değer temizleme yöntemleri modeli güçlendirebilmekle birlikte yanlış uygulandıklarında yanıltıcı sonuçlara yol açabilmektedir. Ayrıca, veri kümeleri

arasındaki farklılıkların genelleme metriklerine net biçimde yansıdığı görülmüştür. Bununla birlikte, ham veri setinin bile MLP benzeri derin öğrenme modelleriyle yüksek başarımları sağlayabildiği saptanmıştır.

Farklı biçimde yapılandırılmış sandbox ortamlarından toplanan test verisi sayesinde, genelleme ve en düşük katlama doğrulukları karşılaştırılarak modellerin kararlılığı incelenmiştir. Farklı ayarlar altında elde edilen başarılı sonuçlar, bu alanda daha gelişmiş konfigürasyonlarla daha yüksek doğruluklara ulaşılabileceğini göstermektedir.

6.5 Derin Öğrenme ve Doğal Dil İşleme Gelecek Çalışmalar

Mevcut Windows olay kayıtları veri setimizi derin öğrenme modellerine uyarlamak için, öncelikle kayıtları zamanı takip eden kısa diziler hâline getirmeyi planlanabilir. Örneğin:

- **Olay Dizileri Oluşturma:** Gerçek zamanlı olay günlüklerini, ardışık 100–200 kayıttan oluşan küçük gruplara bölerek her grubun "birbirini takip eden olaylar" dizisi olarak işlenmesi.
- **Basit Sayısal Gösterimler:** Her bir olay satırında yer alan kayıt numarası, olay türü ve zaman farkı gibi bilgileri sayıya dönüştürüp, bir dizi içinde sıralamak. Böylece, modelimiz "bir olaydan sonra gelen diğer olayların" örüntüsünü görebilecek.
- **Sıra Modeli Eğitimi:** Elde edilen dizileri, zaman içinde en sık rastlanan zararlı davranış kalıplarını yakalamaya uygun olarak tekrar eden sinir ağı (LSTM gibi) veya basit katmanlı konvolüsyonel bir ağ ile eğitmek.
- **Veri Çoğaltma:** Bazı dizilerdeki olaylar atlanıp yerine benzer kayıtlar ekleyerek, modelin "küçük değişimlere" karşı da dayanıklı olması sağlanacak.

Windows olay kayıtlarının doğal dil işleme altyapısıyla derin öğrenme modellerine doğrudan beslenebilir hâle getirilmesi amaçlanabilir. Bunun için öncelikle her bir olay kaydı bir "metin belgesi" gibi kelimelere ayrılarak sayısal vektörlere dönüştürülür; ardından bu vektörler, ardışık zaman dilimleri boyunca üst üste istiflenerek üç boyutlu (örneğin olay sayısı $\times$ zaman adımı $\times$ özellik boyutu) bir veri kümesi oluşturulur. Elde edilen 3 boyutlu yapılar, konvolüsyonel sinir ağları veya zamansal uzantılı değiştirilebilir modelleri gibi modern derin öğrenme mimarilerine doğrudan girilerek, hem

tekil olayların hem de zaman içindeki örüntülerin eş zamanlı olarak yakalanması sağlanabilir. Bu yöntem, sadece metin temelli anomalileri değil, aynı zamanda olayların birbirleriyle olan mekansal-zamansal ilişkilerini de keşfederek siber saldırı tespitinde ek bir seviye proaktiflik ve hassasiyet kazandıracaktır.

6.6 Sonuç ve Tartışma

Bu araştırma, Windows olay kayıtlarının dinamik davranış analizinde ne denli zengin ve değerli bilgiler sağladığını ortaya koymaktadır. Sandbox ortamında izole edilmiş veriler, zararlı ve güvenli yazılımların davranışlarını ayrıntılı şekilde kaydedebilmekte ve elde edilen öznitelikler, makine öğrenmesi modelleri için güçlü ayırıcı girdiler sunmaktadır. Öznitelik iyileştirme sürecinin, tekrarlanabilir bilgileri azaltarak modelin aşırı öğrenme riskini düşürdüğü, dolayısıyla modelin test veri üzerindeki genelleme yeteneğinin artırıldığı gözlemlenmiştir.

Ayrıca, model eğitimi sürecinde kullanılan K-Katlamalı Çapraz Doğrulama yöntemi, modelin tüm veri alt kümeleri üzerinde test edilerek güvenilir performans sonuçları elde edilmesine olanak tanımış, böylece modelin hem eğitim hem de test veri üzerinde tutarlı sonuçlar vermesi sağlanmıştır. Bu da, geliştirilen sistemin gerçek dünya senaryolarında da etkili bir şekilde çalışabileceğini göstermektedir.

Sonuç olarak, elde edilen veriler, Windows olay kayıtlarının dinamik yazılım tespitinde kullanılabilecek önemli bir kaynak olduğunu doğrulamaktadır. Makine öğrenmesi modelleri ile yapılan deneyler, öznitelik iyileştirme stratejilerinin model performansı ve genelleme yeteneği üzerinde kritik etkileri olduğunu ortaya koymuş. Bu da, gelecekteki çalışmalar için sağlam bir temel oluşturmuştur. Gelecekte bu veri setleri üzerinden daha iyi düzenlenmiş veya daha iyi konfigürasyon yapılmış makine öğrenmesi modelleri ile daha fazla ilerleme sağlanabilir.

KAYNAKLAR

- [1] **N. Pachhala and S. Jothilakshmi and B. P. Battula.** A comprehensive survey on identification of malware types and malware classification using machine learning techniques. In *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, pages 1207–1214, Trichy, India, 2021.
- [2] **Aurangzeb, Sana and Aleem, Muhammad.** Evaluation and classification of obfuscated android malware through deep learning using ensemble voting mechanism. *Scientific Reports*, 13, 02 2023.
- [3] **Alhashmi, Asma A. and Darem, Abdulbasit A. and Alashjaee, Abdullah M. and Alanazi, Sultan M. and Alkhaldi, Tareq M. and Ebad, Shouki A. and Ghaleb, Fuad A. and Almadani, Aloyoun M.** Similarity-based hybrid malware detection model using api calls. *Mathematics*, 11(13), 2023.
- [4] **Ce, Li and Cheng, Zijun and Zhu, He and Wang, Lei and Lv, Qiu and Wang, Yan and Li, Ning and Sun, Degang.** Dmalnet: Dynamic malware analysis based on api feature engineering and graph learning. *Computers Security*, 122:102872, 08 2022.
- [5] **Cabrera, Armando and Calix, Ricardo A.** On the anatomy of the dynamic behavior of polymorphic viruses. In *2016 International Conference on Collaboration Technologies and Systems (CTS)*, pages 424–429, 2016.
- [6] **Swarna Priya R.M. and Praveen Kumar Reddy Maddikunta and Parimala M. and Srinivas Koppu and Thippa Reddy Gadekallu and Chiranjil Lal Chowdhary and Mamoun Alazab.** An effective feature engineering for dnn using hybrid pca-gwo for intrusion detection in iomt architecture. *Computer Communications*, 160:139–149, 2020.
- [7] **Saxe, Joshua and Berlin, Konstantin.** Deep neural network based malware

- detection using two dimensional binary program features. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 11–20, 2015.
- [8] **Fangtian Zhong and Pengfei Hu and Guoming Zhang and Hong Li and Xiuzhen Cheng.** Reinforcement learning based adversarial malware example generation against black-box detectors. *Computers Security*, 121:102869, 2022.
- [9] **Qian Li and Qingyuan Hu and Yong Qi and Saiyu Qi and Xinxing Liu and Pengfei Gao.** Semi-supervised two-phase familial analysis of android malware with normalized graph embedding. *Knowledge-Based Systems*, 218:106802, 2021.
- [10] **Taher, Fatma and AlFandi, Omar and Al-kfairy, Mousa and Al Hamadi, Hussam and Alrabaee, Saed.** Droiddetectmw: A hybrid intelligent model for android malware detection. *Applied Sciences*, 13(13), 2023.
- [11] **Steverson, Kai and Carlin, Caleb and Mullin, Jonathan and Ahiskali, Metin.** Cyber intrusion detection using natural language processing on windows event logs. In *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, pages 1–7, 2021.
- [12] **Sinha, Apoorv Kumar and Sai, Shreyas.** Integrated malware analysis sandbox for static and dynamic analysis. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–5, 2023.
- [13] **Amit Sharma and Brij B. Gupta and Awadhesh Kumar Singh and V.K. Saraswat.** Orchestration of apt malware evasive manoeuvres employed for eluding anti-virus and sandbox defense. *Computers Security*, 115:102627, 2022.
- [14] **Lusky, Yehonatan and Mendelson, Avi.** Sandbox detection using hardware side channels. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 192–197, 2021.
- [15] **Cuckoo Sandbox Team.** Cuckoo sandbox: Automated malware analysis. <https://cuckoo.readthedocs.io/>. Eriřim tarihi: 24 Mart 2025.

- [16] **Mills, Alan and Legg, Phil.** Investigating anti-evasion malware triggers using automated sandbox reconfiguration techniques. *Journal of Cybersecurity and Privacy*, 1:19–39, 11 2020.
- [17] **Junyang Qiu and Jun Zhang and Wei Luo and Lei Pan and Surya Nepal and Yang Xiang.** A survey of android malware detection with deep neural models. *ACM Computing Surveys (CSUR)*, 53:1 – 36, 2020.
- [18] **Sethi, Kamalakanta and Kumar, Rahul and Sethi, Lingaraj and Bera, Padmalochan and Patra, Prashanta Kumar.** A novel machine learning based malware detection and classification framework. In *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–4, 2019.
- [19] **Zhang, Zhaoqi and Qi, Panpan and Wang, Wei.** Dynamic malware analysis with feature engineering and feature learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:1210–1217, 04 2020.
- [20] **VirusShare.** Virussshare: Virüs paylaşım platformu. <https://virusshare.com>, 2025. Erişim tarihi: 24 Mart 2025.
- [21] **PortableApps.com.** Portableapps.com: Portable software platform. <https://portableapps.com>, 2025. Erişim tarihi: 24 Mart 2025.
- [22] **Softonic - Download Apps and Software.** <https://www.softonic.com/>. Accessed: 2 February 2025.
- [23] **SourceForge - Open Source Software.** <https://sourceforge.net/>. Accessed: 2 February 2025.
- [24] **CNET - Software Reviews and Downloads.** <https://www.cnet.com/>. Accessed: 2 February 2025.
- [25] **VirusTotal: Free Online Virus, Malware and URL Scanner.** <https://www.virustotal.com/>. Accessed: 2 February 2025.
- [26] **Alecci, Marco and Ruiz Jiménez, Pedro Jesús and Allix, Kevin and Bissyandé, Tegawendé F. and Klein, Jacques.** Androzoo: A retrospective with a

- glimpse into the future. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, pages 389–393, 2024.
- [27] **Lee, Yen-Ting and Ban, Tao and Wan, Tzu-Ling and Cheng, Shin-Ming and Isawa, Ryoichi and Takahashi, Takeshi and Inoue, Daisuke.** Cross platform iot-malware family classification based on printable strings. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 775–784, 2020.
- [28] **AlienVault OTX: Open Threat Exchange platform for threat intelligence.** <https://otx.alienvault.com>. Accessed: 2 February 2025.
- [29] **Shu, Longhui and Dong, Shi and Su, Huadong and Huang, Junjie.** Android malware detection methods based on convolutional neural network: A survey. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(5):1330–1350, 2023.
- [30] **Busch, Julian and Kocheturov, Anton and Tresp, Volker and Seidl, Thomas.** Nf-gnn: Network flow graph neural networks for malware detection and classification, 03 2021.
- [31] **Elnaggar, Rana and Servadei, Lorenzo and Mathur, Shubham and Wille, Robert and Ecker, Wolfgang and Chakrabarty, Krishnendu.** Accurate and robust malware detection: Running xgboost on runtime data from performance counters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(7):2066–2079, 2022.
- [32] **Ring, Markus and Schlör, Daniel and Wunderlich, Sarah and Landes, Dieter and Hotho, Andreas.** Malware detection on windows audit logs using lstms. *Computers Security*, 109:102389, 07 2021.
- [33] **Prasse, Paul and Brabec, Jan and Kohout, Jan and Kopp, Martin and Bajer, Lukas and Scheffer, Tobias.** *Learning Explainable Representations of Malware Behavior*, pages 53–68. 09 2021.
- [34] **Ismail, Setia Juli Irzal and Hendrawan and Rahardjo, Budi and Juhana,**

- Tutun and Musashi, Yasuo.** Malssl—self-supervised learning for accurate and label-efficient malware classification. *IEEE Access*, 12:58823–58835, 2024.
- [35] **Md. Alamgir Hossain and Md Alimul Haque and Sultan Ahmad and Hikmat A. M. Abdeljaber and A. E. M. Eljialy and Abed Alanazi and Deepa Sonal and Kiran Chaudhary and Jabeen Nazeer.** Ai-enabled approach for enhancing obfuscated malware detection: a hybrid ensemble learning with combined feature selection techniques. *International Journal of System Assurance Engineering and Management*, 2024.
- [36] **Malware Bazaar.** <https://bazaar.abuse.ch/>. Accessed: 2 February 2025.
- [37] **Robert J. Joyce and Dev Amlani and Charles Nicholas and Edward Raff.** Motif: A malware reference dataset with ground truth family labels. *Computers Security*, 124:102921, 2023.
- [38] **Muhammad Irfan Yousuf and Izza Anwer and Tanzeela Shakir and Minahil Siddiqui and Maysoon Shahid.** Multi-feature dataset for windows pe malware classification, 2022.
- [39] **Tian, Nan and Tian, Yue and Li, Zizun and Sun, Yizhen and Gu, Haijun and Xiao, Shifeng and Bao, Fei and Hou, Lijuan and Jia, Ming and Lu, Jinci.** Research on usb storage occupancy traceability in windows environment based on log analysis. In *2023 IEEE 7th Information Technology and Mechatronics Engineering Conference (ITOEC)*, volume 7, pages 2156–2160, 2023.
- [40] **Robert J. Joyce and Edward Raff and Charles Nicholas and James Holt.** Maldict: Benchmark datasets on malware behaviors, platforms, exploitation, and packers, 2023.
- [41] **Aboaoja, Faitouri A. and Zainal, Anazida and Ali, Abdullah Marish and Ghaleb, Fuad A. and Alsolami, Fawaz Jaber and Rassam, Murad A.** Dynamic extraction of initial behavior for evasive malware detection. *Mathematics*, 11(2), 2023.
- [42] **Sandborn, Michael and Stoebner, Zach and Weimer, Westley and Forrest, Stephanie and Dougherty, Ryan and White, Jules and Leach, Kevin.** Redu-

- cing malware analysis overhead with coverings. *IEEE Transactions on Dependable and Secure Computing*, 21(4):4133–4146, 2024.
- [43] **Vassilev, Vassil and Ouazzane, Karim and Sowinski-Mydlarz, Viktor and Maosa, Herbert and Nakarmi, Sabin and Hristev, Martin and Radu, Sorin.** Network security analytics on the cloud: Public vs. private case. In *2023 13th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 151–156, 2023.
- [44] **Zulmeika, Annisa Rifky and Bagjasantosa, Marcellinus David Arel and Ismail, Setia Juli Irzal and Hendrawan.** Prevention methods of virtual machine environment recognition by malware. In *2024 18th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pages 1–6, 2024.
- [45] **Mahmoud, Rasmi-Vlad and Anagnostopoulos, Marios and Pastrana, Sergio and Pedersen, Jens Myrup.** Redefining malware sandboxing: Enhancing analysis through sysmon and elk integration. *IEEE Access*, 12:68624–68636, 2024.
- [46] **Songsong Liu and Pengbin Feng and Shu Wang and Kun Sun and Jiahao Cao.** Enhancing malware analysis sandboxes with emulated user behavior. *Computers Security*, 115:102613, 2022.
- [47] **Schnaubelt, Nele Christin.** Virtualisierung von gesicherten apple systemen mit hilfe von virtualbox. Bachelor thesis, Hochschule Mittweida, 2024.
- [48] **Saxena, Deepika and Singh, Ashutosh Kumar.** Osc-mc: Online secure communication model for cloud environment. *IEEE Communications Letters*, 25(9):2844–2848, 2021.
- [49] **Küchler, Alexander and Mantovani, Alessandro and Han, Yufei and Bilge, Leyla and Balzarotti, Davide.** Does Every Second Count? Time-based Evolution of Malware Behavior in Sandboxes. In *NDSS 2021, Network and Distributed Systems Security Symposium*, Virtual, United States, February 2021. ISOC, Internet Society.

- [50] **Tim Verdonck and Bart Baesens and María Óskarsdóttir and Seppe vanden Broucke.** Special issue on feature engineering editorial. *Machine Learning*, 113(7):3917–3928, 2024.
- [51] **Heaton, Jeff.** An empirical analysis of feature engineering for predictive modeling. In *SoutheastCon 2016*, pages 1–6, 2016.
- [52] **Ede, Thijs van and Aghakhani, Hojjat and Spahn, Noah and Bortolameotti, Riccardo and Cova, Marco and Continella, Andrea and Steen, Maarten van and Peter, Andreas and Kruegel, Christopher and Vigna, Giovanni.** Deepcase: Semi-supervised contextual analysis of security events. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 522–539, 2022.
- [53] **Hou, Yiwei and Guo, Lihua and Zhou, Chijin and Xu, Yiwen and Yin, Zijing and Li, Shanshan and Sun, Chengnian and Jiang, Yu.** An empirical study of data disruption by ransomware attacks. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24*, New York, NY, USA, 2024. Association for Computing Machinery.
- [54] **Zheng Lin Chia and Michal Ptaszynski and Fumito Masui and Gniewosz Le-liwa and Michal Wroczynski.** Machine learning and feature engineering-based study into sarcasm and irony classification with application to cyberbullying detection. *Information Processing Management*, 58(4):102600, 2021.
- [55] **Carta, Salvatore and Podda, Alessandro Sebastian and Recupero, Diego Reforgiato and Saia, Roberto.** A local feature engineering strategy to improve network anomaly detection. *Future Internet*, 12(10), 2020.
- [56] **Riabykh, Aleksei and Suleimanov, Ilias and Nagovitsyn, Ilya and Surzhko, Denis and Konovalikhin, Maxim and Koltsova, Olessia.** Entropy-based text feature engineering approach for forecasting financial liquidity changes. *EPJ Data Sci.*, 14(1):17, 2025.
- [57] **Hongfang Zhou and Xiqian Wang and Rourou Zhu.** Feature selection based on mutual information with correlation coefficient. *Applied Intelligence*, 52(5):5457–5474, 2022.

- [58] **Daniel Asante Otchere and Tarek Omar Arbi Ganat and Jude Oghenerurie Ojero and Bennet Nii Tackie-Otoo and Mohamed Yassir Taki.** Application of gradient boosting regression model for the evaluation of feature selection techniques in improving reservoir characterisation predictions. *Journal of Petroleum Science and Engineering*, 208:109244, 2022.
- [59] **V.Nanda Gopal and Fadi Al-Turjman and R. Kumar and L. Anand and M. Rajesh.** Feature selection and classification in breast cancer prediction using iot and machine learning. *Measurement*, 178:109442, 2021.
- [60] **Isaac Kofi Nti and Owusu Nyarko-Boateng and Justice Aning.** Performance of machine learning algorithms with different k values in k-fold cross-validation. *International Journal of Information Technology and Computer Science (IJITCS)*, 13(6):61–71, 2021.
- [61] **Suyeon Yoo and Sungjin Kim and Seungjae Kim and Brent Byunghoon Kang.** Ai-hydra: Advanced hybrid approach using random forest and deep learning for malware classification. *Information Sciences*, 546:420–435, 2021.
- [62] **Akram, Zeeshan and Majid, Mamoon and Habib, Shaista.** A systematic literature review: Usage of logistic regression for malware detection. In *2021 International Conference on Innovative Computing (ICIC)*, pages 1–8, 2021.
- [63] **Gupta, Shashi Kant and Pattnaik, Birajashis and Agrawal, Vineet and Boddu, Raja Sarath Kumar and Srivastava, Archana and Hazela, Bramah.** Malware detection using genetic cascaded support vector machine classifier in internet of things. In *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pages 1–6, 2022.
- [64] **Altyeb Altaher Taha and Sharaf Jameel Malebary.** Hybrid classification of android malware based on fuzzy clustering and the gradient boosting machine. *Neural Computing and Applications*, 33(12):6721–6732, 2021.
- [65] **Karan Gupta and Deepak Kumar Sharma and Koyel Datta Gupta and Anil Kumar.** A tree classifier based network intrusion detection model for internet of medical things. *Computers and Electrical Engineering*, 102:108158, 2022.

- [66] **Saeed Seraj and Siavash Khodambashi and Michalis Pavlidis and Nikolaos Polatidis.** Hamdroid: permission-based harmful android anti-malware detection using neural networks. *Neural Computing and Applications*, 34(18):15165–15174, 2022.
- [67] **Muhammad-Haseeb-Zia and Hussain, Ali and Muhammad-Hamza.** Comparative analysis of random forest and support vector machine classifiers for unjustified malware detection of android devices data consuming smote and roc-auc metrics. In *2024 Horizons of Information Technology and Engineering (HITE)*, pages 1–4, 2024.
- [68] **Zheng, Hongye and Wang, Bingxing and Xiao, Minheng and Qin, Honglin and Wu, Zhizhong and Tan, Lianghao.** Adaptive friction in deep learning: Enhancing optimizers with sigmoid and tanh function. In *2024 IEEE 6th International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pages 809–813, 2024.