|  |  |
| --- | --- |
|  | **T.C.**  **KIRŞEHİR AHİ EVRAN ÜNİVERSİTESİ**  **MHENDİSLİK MİMARLIK FAKÜLTESİ**  **BİLGİSAYAR MÜHENDİSLİĞİ**  **LİSANS** |

**Yapay Zeka Destekli Web Application Firewall**

**202511017**

**Kutluhan AZAFLI**

**LİSANS / DERS PROJESİ**

**KIRŞEHİR**

**2023**

# KULLANILAN DİL, PLATFORM VE VERİSETİ

Proje Anaconda üzerinde çalışan, Python’un 3.11.5 sürümü kullanarak yazılmıştır.

Veriseti internette pek çok kaynaktan toplanan içeriklerin derlenip düzenlenmesi ile oluşturulmuştur. Veriseti iki sütundan oluşmaktadır. Bu sütunlardan ilki “payload” ikincisi ise “label” kısmıdır. Bu sütunları açıklamak gerekirse “payload” kısmı web uygulamasına gönderilen paketlerin payload kısmını içermektedir. Bu kısım siteye gönderdiğimiz requestteki talebimizi belirten kısımdır. Örneğin Google’da bir arama yaptığımızda gönderdiğimiz requesti incelersek:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

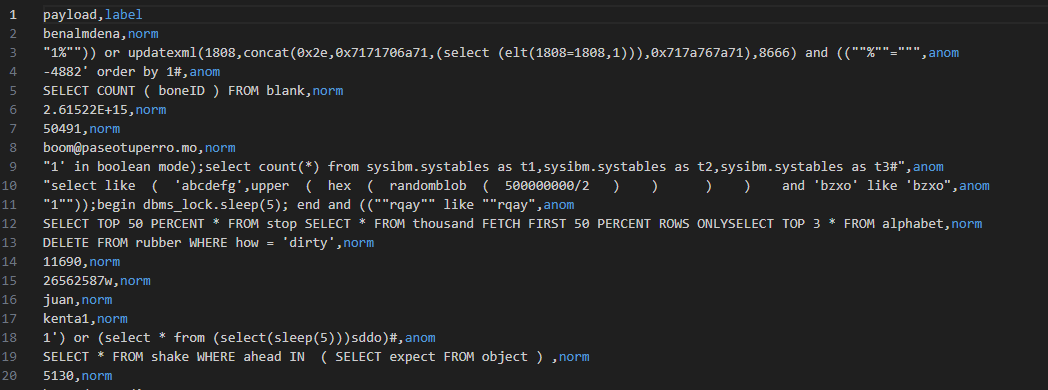
giden paketin payload kısmındaki içeriği burada görmek mümkündür.

Payload kısmındaki verileri incelersek, Zararlı SQL, Dizin Atlama, Siteler Arası Betik Çalıştırma gibi normal bir kullanıcıdan gelmesi beklenmeyen içerikler ve normal bir kullanıcıdan gelmesini beklediğimiz isim soyad, mail adresleri, sayısal veriler gibi veriler kullanılmaktadır.

Burada bir farklılık olarak verisetine zararlı olmayan SQL sorguları da eklenerek bunlar “norm” olarak işaretlenmiştir.

“label” kısmı ise gönderilen requestteki payload’ın zararlı ya da güvenli olduğunu işaretlemek amaçlı kullanılmıştır. Anomali içeren payloadlar “anom”, normal payloadlar ise “norm” olarak işaretlenmiştir.

Veriseti Örneği:



# PROBLEMİN AÇIKLANMASI VE ŞEMATİK GÖSTERİMİ



## 2.1. Problem

Dijital dünya, gittikçe karmaşık hale gelirken, web uygulamaları, şirketlerin iş süreçlerini yönetmelerinde ve müşterilere hizmet sunmalarında kritik bir rol oynamaktadır. Ancak, bu sürekli dijital evrim, siber tehditleri de beraberinde getiriyor. Web uygulamalarının güvenliği, siber saldırılardan korunmak ve hassas verileri korumak için kritik bir öneme sahiptir. Bu noktada, geleneksel güvenlik önlemleri yetersiz kalmaktadır. Yapay zeka destekli Web Application Firewall (WAF) ise bu sorunun üstesinden gelmede yeni bir dönemin kapılarını aralamaktadır.

Geleneksel WAF çözümleri genellikle belirli kurallar ve imza tabanlı yaklaşımlarla çalışır. Ancak, bu yaklaşım, hızla evrilen siber tehditlere ve karmaşık saldırı stratejilerine karşı tam anlamıyla etkili olamaz. Özellikle, modern siber saldırganlar, gelişmiş saldırı teknikleri ve otomatik araçlar kullanarak güvenlik sistemlerini atlamak için sürekli olarak yeni yöntemler geliştirmektedir.

## 2.2. Şematik Gösterim

A diagram of a software company

Description automatically generated

# VERİ YAPISI KATALOĞU

## 3.1. Veri Yapıları

1. * 1. **DataFrame (df)**

Pandas kütüphanesini kullanarak CSV dosyasından okunan veriyi temsil eden DataFrame.

* + 1. **Model Eğitim ve Test Verileri (X\_train, X\_test, y\_train, y\_test)**

Veri setinin eğitim ve test verilerini içeren değişkenler.

## Sınıflar

* + 1. **Flask Uygulaması (app)**

Web uygulamasını temsil eden Flask sınıfı.

route('/analyze\_request', methods=['POST']), HTTP POST isteğine yanıt veren metoddur. Gönderilen istekler bu web sayfası ile karşılanır ve analiz edilir.

## Metodlar

* + 1. **Zararlı İçerik Kontrol Metodu (is\_malicious)**

Verilen metni analiz ederek zararlı içerik içerip içermediğini kontrol eden metot.

* + 1. **Model Eğitme Metotları (fit())**

Veri seti üzerinde farklı modelleri eğiten metotlar.

* + 1. **Model Tahmin Metotları (predict())**

Verilen metni kullanarak gelen isteği analiz eden ve sonucu döndüren metotlar.

* + 1. **Confusion Matrix Görselleştirme Metotu (ConfusionMatrixDisplay())**

Confusion matrix'i görselleştiren metot.

* + 1. **Eğitim Süreleri Görselleştirme Metodu(plt.bar())**

Model eğitim sürelerini görselleştiren metot.

* + 1. **Web Sunucu Başlatma Metodu (app.run(port=5000))**

Flask uygulamasını belirtilen port üzerinde başlatan metot.

# ELDE EDİLEN SONUÇLAR



## Farklı Yapay Zeka Modelleri ile Confusion Matrix Değerlendirmesi

Dört farklı model ile eğitilmiş olan veri setinin doğruluk tahmin oranları confusion matrixde görselleştirilmiştir. Bunlar sırayla Multinomial Naive Bayes, Logistic Regression, Support Vector ve Random Forest sınıflandırıcıları ile yapılmıştır. Bu sınflandırıcılardan önce tüm datalar TfidfVectorizer() ile TF-IDF özellik matrisine dönüştürülmüştür.

A blue squares with white text

Description automatically generated

A diagram of a logistic regression confusion matrix

Description automatically generated

A diagram of a diagram

Description automatically generated with medium confidence

A blue squares with white text

Description automatically generated

## Model Eğitim Süreleri

Modelin eğitimi sürecinde geçen zaman kaydedilmiş ve bu zamanlar matplotlib ile görselleştirilmiştir.

A graph with different colored rectangles

Description automatically generated

## Modellerin Tahmin Süreleri

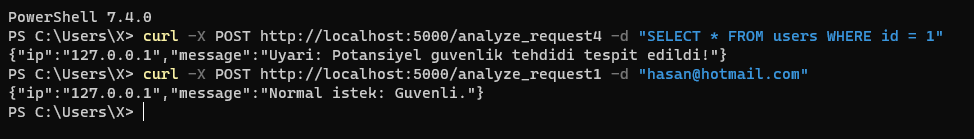
Modellere verilen 6214 test verisinin işlenme ve bir sonuç çıkarma süreleri gözlemlenmiş ve matplotlib aracılığı ile görsellenmiştir.

A graph with red and green squares

Description automatically generated

## Modellerin Gerçek Zamanlı Denenmesi

Modeller için ayrı ayrı oluşturulan test sayfalarına Curl aracılığı ile atılan istekler ile, bu isteklerin anlık olarak izlenmesi sağlanmıştır. Aynı zamanda gönderilen requestlerde göndericinin IP adresi de gerektiğinde kayıt altına alınabileceği bir yapı kurulmuştur.



# KAYNAKLAR

* <https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/>
* <https://scikit-learn.org/stable/user_guide.html>
* <https://book.hacktricks.xyz/pentesting-web/sql-injection>
* <https://book.hacktricks.xyz/pentesting-web/xss-cross-site-scripting>
* <https://book.hacktricks.xyz/pentesting-web/xpath-injection>
* <https://seaborn.pydata.org/>