DDOS Detection

Ad Soyad: Emin Hallak
Numara: 22040301163
Bölüm: Yazılım Mühendisliği
Istanbul Topkapı Üniversitesi
Istanbul/Türkiye

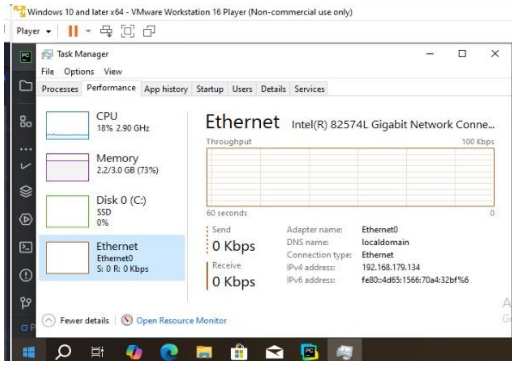
***Abstract*—Günümüzde ülkeler için askeri, iktisadi, idari vb. güçlerin önemli olmasından ziyade artık yaşadığımız teknoloji çağında siber güç devletler için oldukça önemli ve büyük bir rol oynar oldu [1], özellikle geçtiğimiz son 20 yılda oldukça önemli bir yere sahip oldu. Nice bir ülkenin siber gücü zayıf olursa ülkeye karşı siber saldırıların gerçekleşmesine sebep olur ki bu saldırılar devleti mali yönden ve O devletin halkının güvenliğini tehlikeye atabilir. Siber saldırılar yalnızca maddi hasara yol açmakla kalmaz, aynı zamanda uluslararası düzeyde de siyasi krizlere sebep olabilir. Örneğin, devlet kurumlarına veya kritik altyapılara enerji, sağlık, iletişim sistemleri gibi yönelik saldırılar, ülkenin egemenliğini tehdit edebilir. Aynı zamanda da bu şirketler için geçerlidir. Bir şirket kendisini siber yönden güvenli tutmazsa oradaki görevlilerin kişisel bilgileri ve şirketin özel verilerin sızdırılmasına neden olur. Böyle bir şeyin gerçekleşmesi hem görevlilerin şirketten ayrılmasına hem O şirketin batmasına neden olur. Bu yazıda siber güvenliğin bir başlığı olan Ddos Detection ele alınmıştır. Projede yapay sinir ağları teknolojileri kullanılıp bezer yöntemlerin derin öğrenme uygulamalarında başarılı sonuçlar elde etmiştir [2]. Local'de Ddos saldırı yaparak kendi veri Set'imizi oluşturup basit bir yapay sinir ağları modeli ile eğittirerek ve Sigmoid aktivasyon fonksiyonu kullanarak tamamlanmıştır. Eğittirme sonucunda yaklaşık olarak 17 bin veri girişi için (9 bin Ddos 8 bin Normal) trafiği için %97 accuracy sonucu elde edilmiştir. Daha sonra modeli kaydedip implementasyona geçerek Real-Time (gerçek zamanlı) bir saldırı tespit etme sistemi gerçekleştirilmiştir.**

1. DATASET

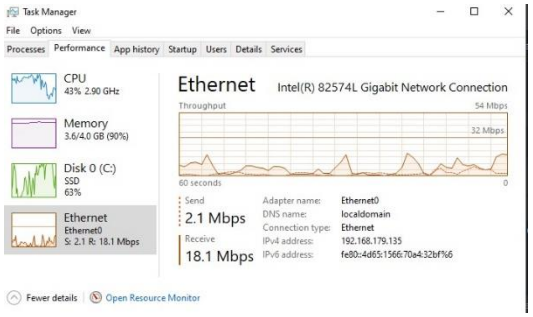
A. *Dataset hakkında bilgiler*

Bahsettiğimiz gibi eğittirme ve doğrulama aşamalarında kendi dataset'imizi oluşturduk. VMware Workstation'ı kullanarak toplamda 4 sanal makine kuruldu (1 windows, 3 kali linux). Ardından kurmuş olduğumuz sanal windows üzerinde Pycharm, Wireshark, Python ve gerekli olan tüm kütüphaneleri kuruldu. Daha sonra Pyshark kütüphanesini kullanarak internet trafiğini Python aracılığıyla CSV dosyasında kaydedildi. Şimdi ise dataset'i hakkında daha çok detaylı bilgiler, içinde bulunan veriler ve işleyişini ele alalım.

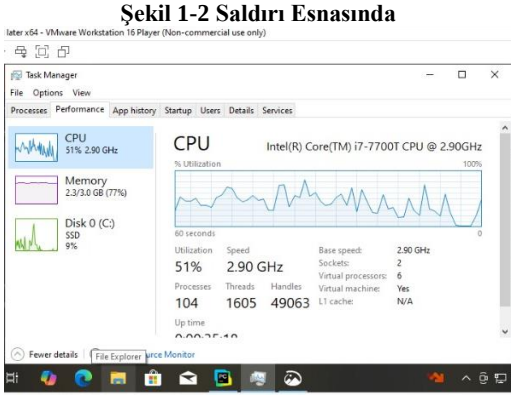
Oluşturmuş olduğumuz dataset 3 aşamadan geçmiştir. İlk aşama pyshark kütüphanesini kullanarak Real-Time internet trafiğini çekip CSV dosyasında yazan kodu geliştirme, internet trafiği ile birlikte saldırı olup olmadığını etiketlemek için "label" sütünü eklendi. İkinci aşama ise hem saldırı hem de normal ağ trafiği esnasında verileri toplayıp bir CSV dosyasında işlemektir. Sütünlardaki "label" değeri 1'e ayarlayıp kali linux sanal makinelerden saldırıp kodu çalıştırdık. Şekil 1.1 saldırı öncesi, saldırı esnası ve saldırıdan sonra makinenin durumunu inceleyebilirsiniz. Ardından kodu kapatıp ilk CSV dosyasını elde ettik. Daha sonra "label" değerini 0'a ayarlayıp tekrar çalıştırıldı ve bu sefer internet üzerinden stream dosya indirme siteler arası gezinme gibi işlemler yapıldı ve normal internet trafiği elde edildi. Son aşama olarak elde edilen CSV dosyalarını tek bir CSV dosyasında toplamaktı. Verilerin çoğunluğu TCP protokolü üzerinden gerçekleştirilmiştir.



Şekil 1-1 Saldırı Öncesi



Şekil 1-2 Saldırı Esnasında



Şekil 1-3 Saldırıdan Sonra Makinenin İnternete Erişmemesi

B. Dataset'in içeriği

Dataset'te bulunan kolon sayısı 19'tür ve şunlardır (Source Ip, timestamp, Dst Port, Protocol, Flow Duration, Tot Fwd Pkts, Tot Bwd Pkts, TotLen Fwd Pkts, TotLen Bwd Pkts, Flow Byts/s, Flow Pkts/s, Pkt Len Mean, FIN Flag Cnt, SYN Flag Cnt, RST Flag Cnt, PSH Flag Cnt, ACK Flag Cnt, URG Flag Cnt, Label) bu kolonları tek tek inceleyelim ve ne anlama geldiğine bakalım:

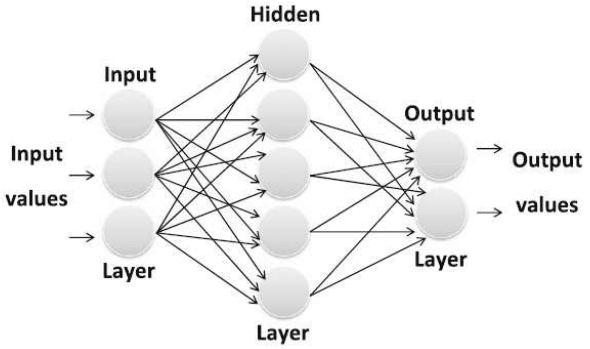
- Source Ip**:** Sunucuya, bilgisayara veya makineye gelen isteklerin IP adresidir.
- Timestamp: Unix formatında trafik akışının başladığı zaman.
- Protocol**:** Ağda kullanılan iletişim protokolüne karşı sayısal değeri belirtir (6: TCP, 17: UDP vb.).
- Dst Port: Ağ trafiğinin gerçekleştiği istekler hangi hedef port numarasına gönderildiğini ifade eder.
- Flow Duration: saniye cinsinden trafik akışının süresini ifade eder.
- Tot Fwd Pkts (Total Forward Packets): Kaynaktan hedefe gönderilen packet sayısını ifade eder.
- Tot Bwd Pkts (Total Backward Packets): Hedeften kaynağa dönen packet sayısını ifade eder.
- TotLen Fwd Pkts (Total Length of Forward Packets): bayt cinsinden kaynaktan hedefe gönderilen tüm packetlerin boyutunu ifade eder.
- TotLen Bwd Pkts (Total Length of Backward Packets): bayt cinsinden hedeften kaynağa dönen tüm packetlerin boyutunu ifade eder.
- Flow Byts/s (Flow Bytes per Second): Trafik akışında saniye başına aktarılan bayt miktarını ifade eder.
- Flow Pkts/s (Flow Packets per Second): Trafik akışında saniye başına aktarılan packet sayısını ifade eder.
- Pkt Len Mean (Packet Length Mean): packetlerin ortalama boyutunu ifade eder.
- FIN Flag Cnt: TCP bağlantılarını kapatmak için kullanılan FIN flag sayısını ifade eder.
- SYN Flag Cnt: Bağlantı başlatmak için kullanılan SYN flag sayısını ifade eder.
- RST Flag Cnt: Bağlantıyı sıfırlamak için kullanılan RST flag sayısını ifade eder.
- PSH Flag Cnt: Veriyi anında iletmek için kullanılan PSH flag sayısını ifade eder.
- ACK Flag Cnt: Packetlerin doğru ve eksik bir şekilde teslim ediliğini onaylamak için kullanılan ACK flag sayınını ifade eder.
- URG Flag Cnt: Acil veri taşınğını gösteren URG flag sayısını ifade eder.

Not: eğittirmede bazı kolonlar göz ardı edilmiştir. network trafiğinde bilgisi alınmadığı için ve bazılarını sadece yeni özellikleri türetmek için kullanıldı.

2. YAPAY SİNİR AĞLARI

A. MLP Classifier

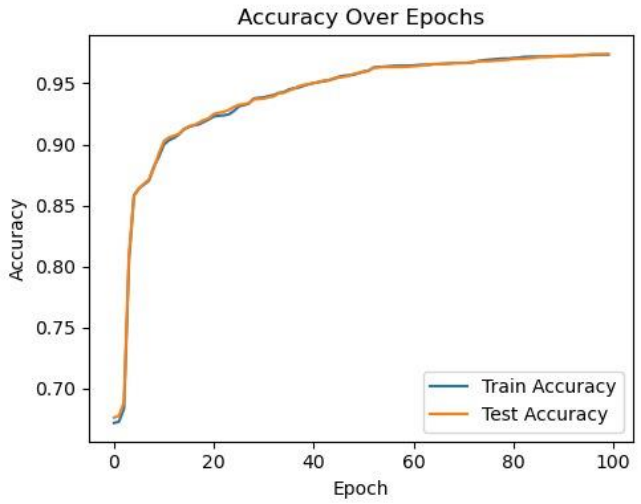
MLP Classifier yapay sinir ağlarının özel bir türüdür. 32 nöronlu gizli katman ve Sigmoid aktivasyon fonksiyonu kullanıldı. Sigmoid aktivasyon fonksiyonu ikili sınıflandırma(0 ve 1) için kullanıldı. Şekil 2-1 de MLP Classifier örneğini inceleyebilirsiniz.



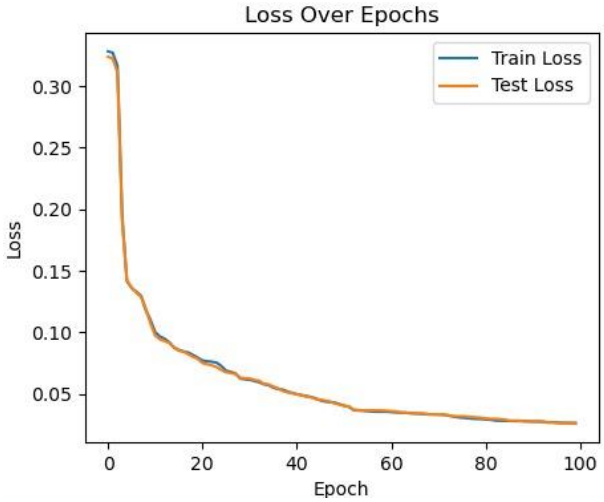
Şekil 2-1 MLP Classifier Örneği

B. Eğittirme ve test etme

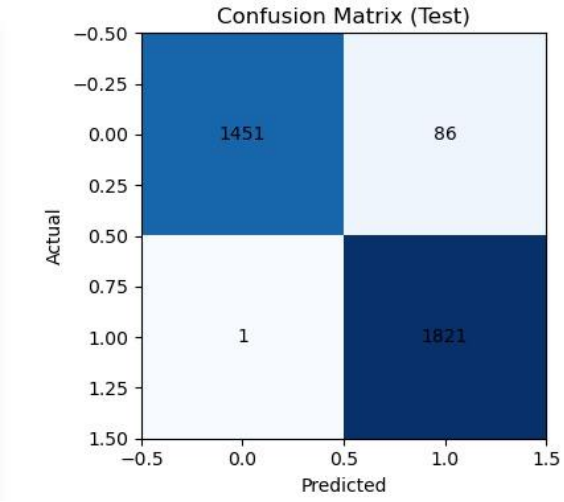
- Öncelikle datasette bulunan verilerin tamamı çekilerek gereksiz ve eksik veriler temizlenmiştir. Bu işlemin modelin daha doğru ve sağlıklı sonuçlar vermesi için önemlidir.
- Ardından yeni Source IP ve Timestamp özellikleri kullanarak "Time_diff" ve "Rolling_avg" özellikleri türetildi. "Time_diff" aynı IP adresten gelen iki packetin arasındaki zaman farkı. "Rolling_avg" aynı IP adresten gelen son 5 packetin "Time_diff" ortalamasıdır.
- Modelin giriş özellikleri StandardScaler normalize edildi. Ardından Sklearn kütüphanesinin "train_test_spilt" fonksıyonunu kullanarak datasetimizi %80 train %20 test oranlarına bölündü.
- Sonrasında 32 nöronlü tek gizli katmanlı MLP Classifier modeli oluşturuldu. Sigmoid ve ve SGD (Stokastik Gradyan İnişi) algoritması kullanılmıştır.
- Öğrenme hızı 0,001 olarak belirlenmiştir. Warm start özelliği kullanarak 100 epoch boyunaca eğittirilip her 10 epoch'ta ekrana accuracy ve loss bilgilerini ekrana yazmasını sağlanmıştır.
- Eğittirme süreci bittikten sonra Matplotlib kütüphanesini kullanarak Şekil 2-2'de modelin train ve test accuracy grafiğini, Şekil 2-3'de modelin train ve test loss grafiğini ve Şekil 2-4'de confusion matrix'i görselleştirilmiştir. En sonda şekil 2-5'de son epoch'ların accuracy ve loss değerini inceleyebilirsiniz.
- Son olarak scaler ve modeli Real-Time Detection için kaydedildi.



Şekil 2-2 Modelin Train Ve Test Accuracy Grafiği



Şekil 2-3 Modelin Train Ve Test Loss Grafiği



Şekil 2-4 Confusion Matrix

```
Epoch 50 | Train Loss: 0.0405 | Train Acc: 0.9595 | Test Loss: 0.0402 | Test Acc: 0.9598
Epoch 60 | Train Loss: 0.0353 | Train Acc: 0.9647 | Test Loss: 0.0360 | Test Acc: 0.9640
Epoch 70 | Train Loss: 0.0332 | Train Acc: 0.9668 | Test Loss: 0.0333 | Test Acc: 0.9667
Epoch 80 | Train Loss: 0.0293 | Train Acc: 0.9707 | Test Loss: 0.0301 | Test Acc: 0.9699
Epoch 90 | Train Loss: 0.0275 | Train Acc: 0.9725 | Test Loss: 0.0277 | Test Acc: 0.9723
```

Şekil 2-5 Son Epoch'ların Detaylı Bilgileri

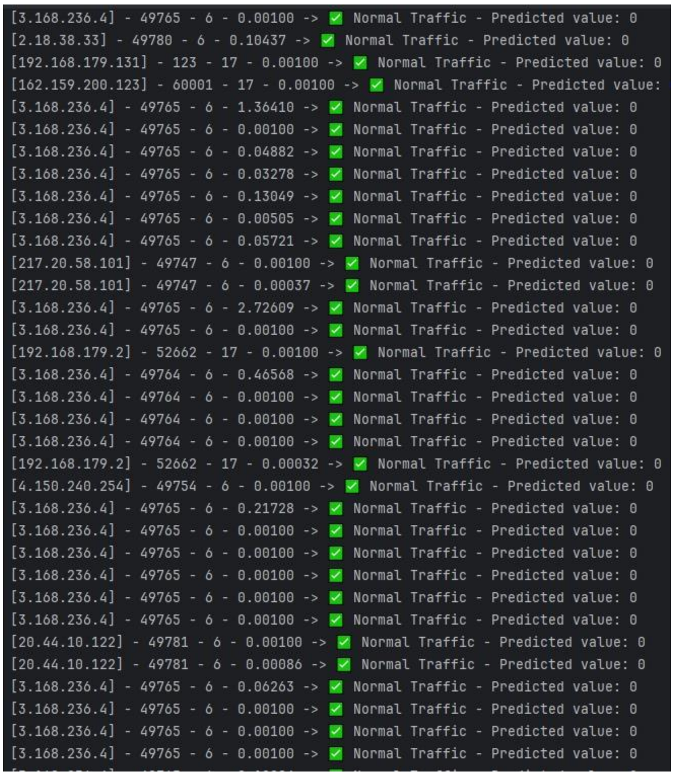
Bu confusion matrisinde sistemin tahmin etme duyarlılığını görüntülü bir şekilde göstermektedir. Gördüğümüz gibi model çoğu yerde ve büyük olasılıkta doğru tahmin sonuçları verebileceğini gösteriyor.

Sonuç olarak %97 bir validation accuracy ve %27 validation loss değerleri ile eğittirmeyi büyük bir başarı oranıyla tamamlamış model sonuçları kısmını bitirmiş oluyoruz. Şimdi ise modeli implement aşamasına geçebiliriz.

3. REAL-TİME SALDIRI TESPİTİ

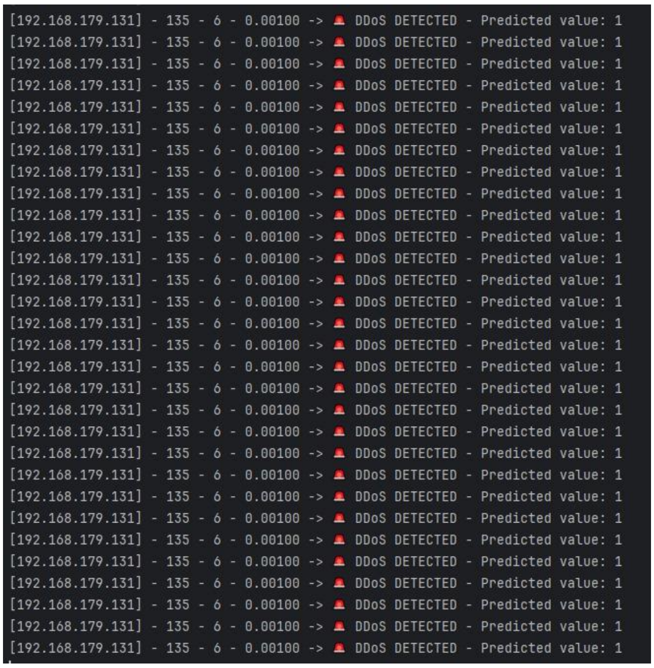
Modeli Real-Time olarak deneme amacıyla ilk önce Wireshark'in Tshark interface'ini kurduk. pythonda wireshark uygulaması kullanarak ağ trafiği çekip modele veren bir kod geliştirdik. İlk önce kaydetmiş olduğumuz model ve scaler dosyalarını pickle'ı kullanarak yükledik. Sonrasında pyshark kütüphanesini kullanarak makinemizin ağ arayüzü üzerinde gerçekşekleşen ağ trafiğini yakalama işlemi sağladık. DDOS saldırı tespitide saldırı sadece tek yönlü gerçekleştiğinden dolayı (pek çok makineden bizim makineye packet gönderimi) bizim makinemizden giden packetleri kontrol etmemiz gerekmediğinden dolayı Socket kütüphanesini kullanarak makinemizin IP adresini dinamik olarak kaydedip tespit aşamasına gelmeden önce geçmesini sağladık. En sonda geliştirmiş olduğumuz modelde kullandığımız packetlerin özellikleri çıkartıp gerekli olan hesaplamaları yaparak elde edilen değerlerden bir Dataframe oluşturarak modele verdik En sonda da Kaynak IP adresi, hedef Port numarası, protocol'ün sayısal değeri, flow duration ve sonucu yazdırdık.

Real-Time detection kodumuz hazır olduğuna göre normal trafik ve local'de yapılan DDOS saldırısı esnasında modelimizi denedik. İlk önce makinemiz üzerinde VMware Kullanarak 4 farklı sanal makine (3 tane kali linux ve 1 tanesi windows) olarak kurduk. Kurmuş olduğumuz windows üzerinde Python ve gerekli kütüphaneleri kurduk. Ardından kodumuzu ve modelimizi sanal windows üzerinde doğru bir şekilde çalıştığını kontrol ettik. Asıl amacımız geliştirmiş olduğumuz modeli canlıda test etmek olduğundan windows firewall'u kapattık. Daha sonrasında kurmuş olduğumuz sanal kali linux makinelerinde Hping3 tool'u kullanarak sanal windows makinesine saldırı gerçekleştirdik. Şekil 3-1 ve Şekil 3-2'de Real-Time saldırı gerçekleştirmeden önce ve gerçekleştirdikten sonra modelin gerçek zamanlı denemede sonuçlarını inceleyebilirsiniz.



```
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[2.18.38.33] - 49780 - 6 - 0.10437 -> ✅ Normal Traffic - Predicted value: 0
[192.168.179.131] - 123 - 17 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[162.159.200.123] - 60001 - 17 - 0.00100 -> ✅ Normal Traffic - Predicted value:
[3.168.236.4] - 49765 - 6 - 1.36410 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.04882 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.03278 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.13049 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00505 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.05721 -> ✅ Normal Traffic - Predicted value: 0
[217.20.58.101] - 49747 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[217.20.58.101] - 49747 - 6 - 0.00037 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 2.72609 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[192.168.179.2] - 52662 - 17 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49764 - 6 - 0.46568 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49764 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49764 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49764 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[192.168.179.2] - 52662 - 17 - 0.00032 -> ✅ Normal Traffic - Predicted value: 0
[4.150.240.254] - 49754 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.21728 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[20.44.10.122] - 49781 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[20.44.10.122] - 49781 - 6 - 0.00086 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.06263 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
[3.168.236.4] - 49765 - 6 - 0.00100 -> ✅ Normal Traffic - Predicted value: 0
```

Şekil 3-1 Gerçek Zamanlı Normal Trafik Tespiti



```
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
[192.168.179.131] - 135 - 6 - 0.00100 -> 🚨 DDoS DETECTED - Predicted value: 1
```

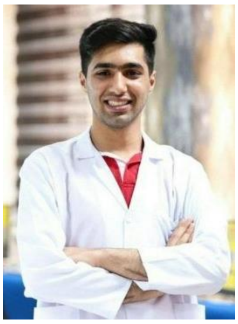
Şekil 3-2 Gerçek Zamanlı DDOS Saldırısı Tespiti

Dikkat ettiğiniz gibi DDOS packetleri zaman ve byte açısından normal packetlere göre daha düşük değerlere sahiptir. DDOS saldırılarında amaç aşırı fazla istek göndererek sunucuyu çökertmek olduğu için packetlerin gönderim hızı ve packet gönderme sıklığı aşırı yüksek iken ve büyüklüğü mümkün olduğu kadar az olması gerekmektedir. Görmüş olduğunuz gibi packetlerin değerlerine göre modelimiz DDOS Tespit işlemi yapabilmektedir.

REFERENCES

[1] Bhuyan, Monowar H., Dhruba Kumar Bhattacharyya, and Jugal K. Kalita. "Network anomaly detection: methods, systems and tools." Ieee communications surveys & tutorials 16.1 (2013): 303-336.

[2] Cil, A. E., Yildiz, K., & Buldu, A. (2021). Detection of DDoS attacks with feed forward based deep neural network model. Expert Systems with Applications, 169, 114520.



Emin Hallak, kendisi 2004 doğumludur. Meslek lisesinde bilişim teknolojileri bölümünden mezun oldu. Istanbul Topkapı Üniversitesinde üçüncü sınıf yazılılm mühendisliği okumaktandır. FullStack geliştiricisidir. Golang, Vuejs, Typescript ve Postgres sql teknolojilerine hakim ve o teknolojileri kullanarak projeler geliştirmektedir. Derin öğrenmede human identification based hand dorsal vein patterns ve İnternet trafiğinde Anomaly detection. Kotlinde yüz tanıma tabanlı öğrenci yoklama sistemini geliştirmektedir.

Github Profile Link: https://github.com/EminHALLAK

Github Project Link: https://github.com/EminHALLAK/DDOS-Detection-Using-MLP