

## **İÇİNDEKİLER**

I.	Proje Özeti ve Amaç.....	1
II.	Veri Kaynağı ve Veri Toplama .....	2
III.	Hedef Özellik Sayısı ve Şekillendirme .....	2
IV.	Çıkarılan Özellikler (Feature Set).....	3
V.	Veri Temizleme ve Ön İşleme.....	4
VI.	Model Mimarisi: LSTMClassifier.....	4
1)	LSTM Katmanı .....	4
2)	Çıkış Katmanı .....	4
VII.	Eğitim Süreci.....	5
1)	Veri Bölme: .....	5
2)	Loss, Optimizer ve Early Stopping : .....	5
VIII.	Model Performansı ve Metrikleri .....	5
	Confusion Matrix ve Classification Report.....	5
IX.	Gerçek Zamanlı Anomali Tespiti .....	8
1)	Buffer Mekanizması .....	8
	Detect_anomaly() fonksiyonu gelen tek satırı: .....	8
	KAYNAKÇA .....	9

## **I. Proje Özeti ve Amaç**

Bu projede amaç, PCAP/PCAPNG formatındaki ağ trafiği kayıtlarından öznitelik (feature) çıkararak denetimli bir LSTM tabanlı saldırı/anomali tespit modeli eğitmek ve gerçek zamanlı aksıta şüpheli davranış görüldüğünde kullanıcıya uyarı üretmektir. Sistem iki ana bileşenden oluşur:

- Veri hazırlama hattı (PCAP → CSV): Paketlerden istatistiksel özellikler çıkarılır, pencere (window) mantığıyla örnekler oluşturulur ve etiketlenir.
- Modelleme ve izleme: Eğitilen modelin tahmin olasılıklarıyla ROC/PR gibi metrikler hesaplanır ve canlı trafikte eşik aşımı durumunda alarm üretir.

## **II. Veri Kaynağı ve Veri Toplama**

### 1) Veri Kaynağı:

Veri kaynağı, ağ trafiğinin pcap / pcapng / cap kayıtlarıdır. PCAP dosyaları, ağ üzerindeki paketleri zaman sıralı şekilde içerir ve bu paketlerden TCP/UDP/ICMP gibi protokol bilgileri, portlar, bayraklar (SYN/FIN/RST) ve zaman damgaları gibi alanlar çıkarılabilir.

### 2) Veri Toplama Yaklaşımı:

Veri iki sınıfaya ayrılır;

- Normal trafik (label=0): Günlük/olağan ağ davranışını temsil eden PCAP kayıtları
- Saldırı trafiği (label=1): Saldırı/anomali içeren PCAP kayıtları (ör. veri sizintisi, tarama, flood vb.)

### 3) Kodda bu etiketleme şu mantıkla yapılır:

- is\_attack=True ise label = 1
- is\_attack=False ise label = 0

### 4) Dosya Organizasyonu:

Kod, PCAP'leri üç farklı kullanım senaryosu ile işleyebilecek şekilde tasarlanmıştır:

- Tek PCAP dosyası işleme
- Bir klasördeki tüm PCAP'leri işleme (alt klasörler dahil)
- Normal ve Saldırı PCAP'lerini iki ayrı klasörden alıp ayrı CSV üretme

Bu sayede veri seti hazırlama aşaması hem küçük testlerde hem de büyük klasör yapılarında ölçeklenebilir hale gelir.

### **III. Hedef Özellik Sayısı ve Sekillendirme**

- Modelin girişinde her satırdan 19 adet sayısal özellik kullanılacak şekilde standartlaştırma yapılır:
- Eğer CSV'deki sayısal sütun sayısı 19'dan fazlaysa, ilk 19'u alınır.
- Eğer 19'dan azsa, eksikler pad\_0, pad\_1, gibi sıfır değerlerle tamamlanır.
- Sonrasında sütunlar tek tip olsun diye feature\_0 feature\_18 şeklinde yeniden adlandırılır.
- Bu yaklaşım, farklı kaynaklardan gelen CSV'lerin model tarafından tutarlı biçimde kullanılmasını sağlar.

### **IV. Çıkarılan Özellikler (Feature Set)**

#### **1) Trafik Hacmi ve Paket Boyutu İstatistikleri:**

- total\_packets: Pencere içindeki paket sayısı
- total\_bytes: Pencere içindeki toplam byte
- avg\_packet\_size: Ortalama paket boyutu
- std\_packet\_size: Paket boyutlarının standart sapması

Saldırı trafigi çoğu zaman normalden farklı paket boyutu/hacim davranışını göstermesi amaçlanmıştır.

#### **2) Protokol Dağılımı:**

tcp\_ratio, udp\_ratio, icmp\_ratio: TCP/UDP oranları

ICMP artışı, UDP flood, TCP ağırlıklı tarama gibi davranışlar ayırt edilebilir.

#### **3) Port Davranışı ve Entropi:**

- unique\_dst\_ports: Hedef port çeşitliliği
- port\_entropy: Hedef portların Shannon entropisi

Port taraması gibi saldırılarda çok sayıda farklı porta gidildiği için hem çeşitlilik hem entropi artabilir.

#### **4) Zaman Tabanlı Özellikler:**

- avg\_packet\_interval: Paketler arası ortalama süre
- std\_packet\_interval: Paketler arası süre sapması
- packets\_per\_second (PPS): Saniyedeki paket sayısı
- PPS'nin anlamı: Belirli bir zaman aralığında ağın ne kadar "yoğun" aktığını gösterir. PPS yükseldikçe aynı sürede daha fazla paket üretilmiş demektir; bu, flood/DDoS/yoğun tarama gibi durumlarda kritik bir göstergedir.

#### **5) TCP Bayrak (Flag) Özellikleri:**

- syn\_count, fin\_count, rst\_count: SYN/FIN/RST sayıları

- syn\_fin\_ratio: syn\_count / (fin\_count + 1)

SYN flood gibi saldırılar SYN davranışını anormal yükseltebilir; RST artışı bağlantı reset paternleri gösterebilir.

## 6) IP Çeşitliliği:

- unique\_src\_ips, unique\_dst\_ips: Kaynak/hedef IP çeşitliliği
- ip\_diversity: kaynak IP çeşitliliğini paket sayısına göre normalize eden oran

Dağıtık kaynaklı saldırılarda (çoklu kaynak IP) çeşitlilik artabilir.

## V. Veri Temizleme ve Ön İşleme

### Temizleme Adımları:

- Sayısal olmayan sütunlar elenir (select\_dtypes(include=[np.number]))
- inf / -inf değerleri NaN'e çevrilir
- NaN değerler 0 ile doldurulur
- Gerekirse tarih/IP/UID gibi model için kullanılmayan sütunlar düşürülür (timestamp, src\_ip, dst\_ip vb.)

Model girişinin tamamen sayısal ve stabil olmasını sağlamaktır.

### Ölçekleme (MinMaxScaler):

- Eğitimden önce veriler MinMaxScaler(feature\_range=(0,1)) ile 0–1 aralığına çekilir. Bu özellikle LSTM gibi ağlarda:
  - Öğrenmeyi hızlandırır,
  - Farklı ölçekteki feature'ların baskınlığını azaltır,
  - loss dalgalanmasını düşürür.
  -

### PCA ile Boyut İndirgeme:

- Ölçeklenmiş veriyi daha az boyuta indirger,
- Açıklanan varyansı %95 seviyesinde tutar,
- Gürültüyü azaltıp genelleme performansını artırmaya yardımcı olabilir.

## VI. Model Mimarisi: LSTMClassifier

Model, PyTorch ile tanımlanmış bir **ikili sınıflandırıcıdır**.

### 1) LSTM Katmanı

- Girdi şekli: (batch, seq\_len, n\_features)
- hidden\_dim=64, num\_layers=2, dropout=0.3
- batch\_first=True: veri "batch" odaklı gelir.

LSTM'in amacı, ardışık 10 satırlık (sequence) davranışını birlikte değerlendirip saldırının paterni olup olmadığını yakalamaktır.

## 2) Çıkış Katmanı

Son zaman adımının hidden state'sı alınır:

- last\_hidden = out[:, -1, :]
- Dropout uygulanır
- Linear(hidden\_dim → 1) ile tek logit üretilir

Bu logit daha sonra sigmoid ile attack olasılığına çevrilir.

Not: PCA sadece train verisi üzerinde fit edilir, test verisine transform uygulanır. Bu, veri sizintisini (data leakage) önlemek için kritiktir.

## 3) Sequence (Zaman Penceresi) Oluşturma Mantığı

Bu projede kritik tasarım: model tek satır değil, ardışık 10 satır bakarak karar verir.

- SEQUENCE\_LENGTH = 10
- create\_sequences\_supervised() fonksiyonu ile:
  - o her sequence: 10 satır feature
  - o sequence etiketi: o 10 satır içindeki etiketlerin maksimumu

Yani pencere içinde 1 tane bile saldırısı varsa o sequence "Attack" sayılır.

Bu yaklaşım, "saldırı kısa süreli olsa bile" pencerede yakalanmasını kolaylaştırır.

## VII. Eğitim Süreci

### 1) Veri Bölme:

Önce tüm normal + saldırısı verisi birleştirilir. Sonra:

- train\_test\_split(test\_size=0.2, stratify=y\_all)
- Stratify ile 0/1 oranı korunur.

Eğitim sırasında ayrıca sequence bazında tekrar train/val split yapılır (val\_ratio=0.2, stratify).

### 2) Loss, Optimizer ve Early Stopping :

- Kayıp fonksiyonu: BCEWithLogitsLoss()
- Optimizasyon: Adam(lr=1e-3)
- Gradient clipping: clip\_grad\_norm\_(max\_norm=1.0) (patlamayı önler)
- Early stopping: patience=3 (3 epoch iyileşme yoksa durur)

Eğitim sırasında her epoch için Train Loss ve Val Loss loglanır; en iyi val loss'a sahip ağırlıklar saklanıp geri yüklenir.

## VIII. Model Performansı ve Metrikleri

Değerlendirme test setinde **sequence bazında** yapılır. Model çıktıları olasılığa çevrilir ve threshold=0.5 ile sınıf kararı üretilir.

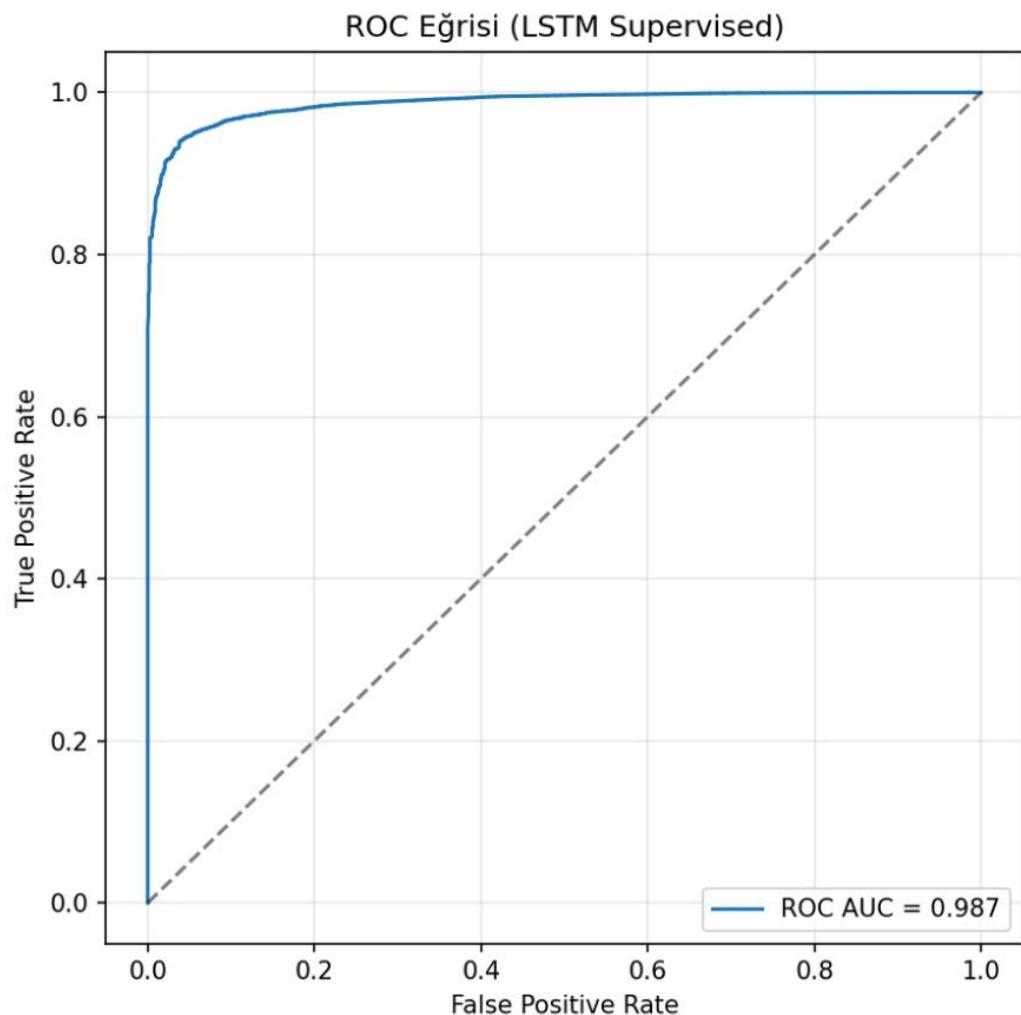
## Confusion Matrix ve Classification Report

Confusion matrix ile:

- **TP:** Saldırıyı doğru yakalama
- **FP:** Yanlış alarm (normal iken saldırıyı demek)
- **FN:** Kaçırılan saldırı
- **TN:** Normalin doğru tanınması

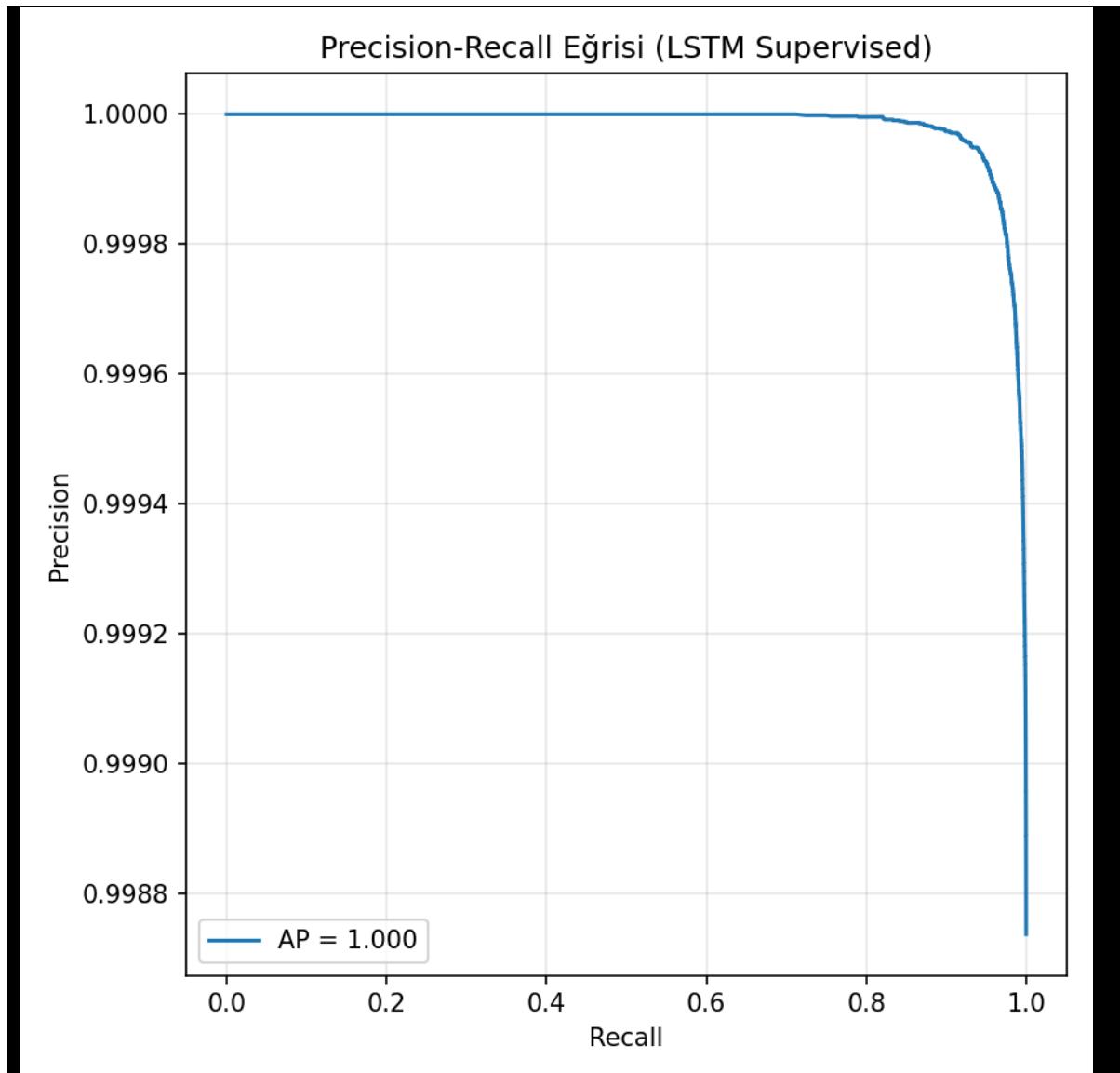
Classification report; precision/recall/f1 değerlerini sınıf bazında verir. Bu metrikler, sistemin “kaç saldırıyı yakaladığı” ve “kaç yanlış alarm verdiği” gibi operasyonel açıdan önemli noktaları gösterir.

ROC Eğrisi ve AUC:



- **ROC AUC = 0.987**
- Bu değer, modelin normal ve saldırıyı sınıflarını ayırmaya başarısının çok yüksek olduğunu gösterir. Eğrinin sol-üst köşeye yakın seyretmesi, düşük yanlış alarm (FPR) ile yüksek

Precision–Recall Eğrisi ve AP:

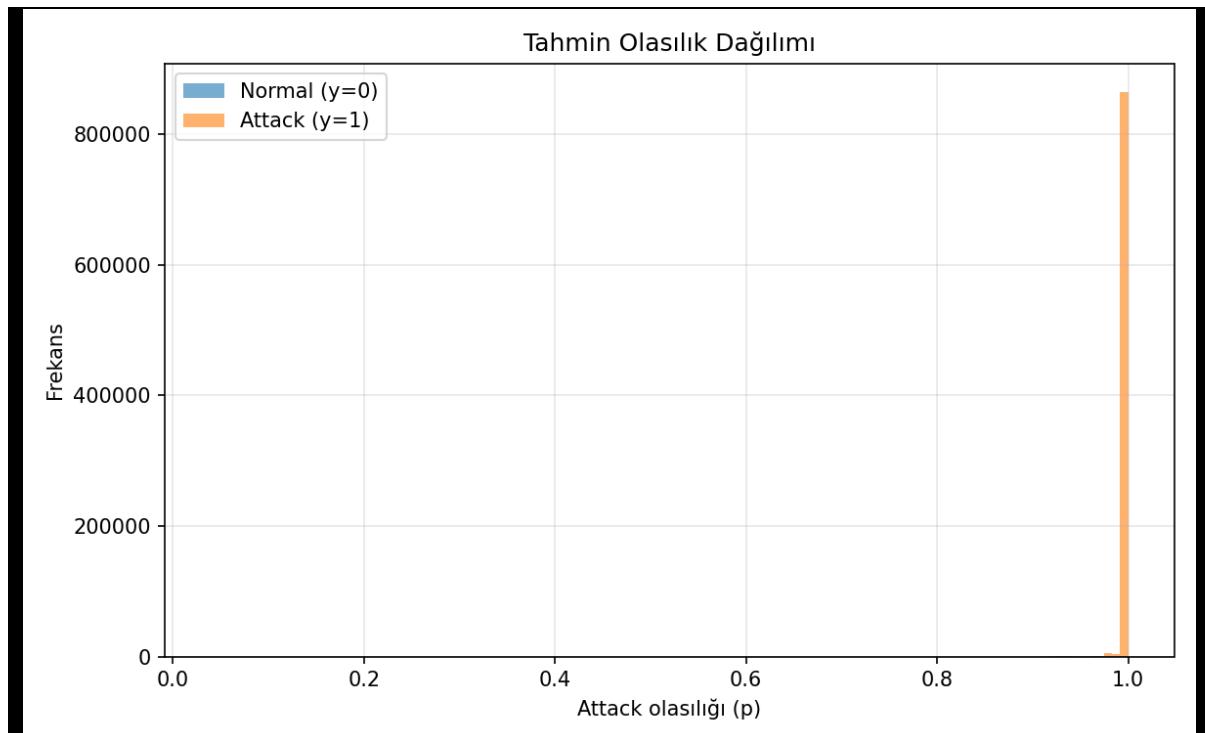


- **AP = 1.000**

PR eğrisi, özellikle saldırı verisinin az olduğu (dengesiz) senaryolarda daha açıklayıcıdır. Paylaşılan grafikte:

Bu, veri seti ve değerlendirme koşullarında modelin saldırı sınıfını ayırmada çok güçlü olduğunu gösterir. (Not: AP'nin 1.000 çıkması bazen veri setinin çok kolay ayrışması, örneklerin benzerliği veya train-test ayrımında sizıntı (data leakage) ihtimali gibi durumlarda da görülebilir. Bu nedenle veri ayırm süreci ayrıca kontrol edilmelidir.)

## Tahmin Olasılık Dağılımı:



Olasılık histogramı, modelin iki sınıfı olasılık uzayında nasıl ayırdığını gösterir:

- Normal sınıf olasılıkları düşük bölgede,
- Attack sınıf olasılıkları 1'e yakın bölgede yoğunlaşıyorsa,

Modelin sınıfları net ayırabildiği anlaşılır. Bu grafik aynı zamanda eşik (threshold) seçiminde pratik bir referans sağlar.

## **IX. Gercek Zamanlı Anomali Tespiti**

### **1) Buffer Mekanizması**

Detect\_anomaly() fonksiyonu gelen tek satırı:

1. Temizler
2. scalar uygular
3. PCA varsa dönüştürür
4. deque buffer'a ekler

Buffer uzunluğu SEQUENCE\_LENGTH=10 olunca LSTM'e verilir ve attack olasılığı üretilir:

- Prob = sigmoid(logit)
- prob >= threshold ise anomali.

```
Paket: 3200 | Anomali: 0 | PPS: 15.3 | Buffer: 100u
=====
⚠ ANOMALİ TESPİT EDİLDİ!
=====
Zaman: 2025-11-22 03:58:49
Skor: 0.0438
Tip: Veri Sızıntısı
PPS: 1533.0
SYN Count: 0.0
Unique Ports: 3.0
=====
```

- PPS (Packets Per Second): Saniyede geçen paket sayısıdır.
- 1533 PPS, ilgili pencerede ağır çok yoğun aktığını gösterir. Bu tür yoğunluk artışları bazı saldırı tiplerinde (flood, tarama, hızlı veri akışı) anlamlı bir sinyal olabilir; fakat kesin yorum için normal trafikteki PPS dağılımıyla kıyas yapılmalıdır.

## KAYNAKÇA

1. Chandola, V., Banerjee, A., & Kumar, V. (2009). *Anomaly detection: A survey*. ACM Computing Surveys, 41(3), 1–58.
2. Patcha, A., & Park, J. M. (2007). *An overview of anomaly detection techniques: Existing solutions and latest technological trends*. Computer Networks, 51(12), 3448–3470.
3. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). *A survey of network anomaly detection techniques*. Journal of Network and Computer Applications, 60, 19–31.
4. Lazarevic, A., Kumar, V., & Srivastava, J. (2005). *Intrusion detection: A survey*. Managing Cyber Threats, Springer.
5. Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). *Long short term memory networks for anomaly detection in time series*. Proceedings of the 23rd European Symposium on Artificial Neural Networks (ESANN).
6. Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural Computation, 9(8), 1735–1780.
7. Kim, S., Song, S., Kim, Y., & Lee, J. (2018). *Anomaly detection in network traffic using LSTM-based autoencoder*. IEEE International Conference on Big Data and Smart Computing.
8. Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2019). *DeepAnT: A deep learning approach for unsupervised anomaly detection in time series*. IEEE Access, 7, 1991–2005.
9. Ergen, T., & Kozat, S. S. (2018). *Unsupervised anomaly detection with LSTM neural networks*. IEEE Transactions on Neural Networks and Learning Systems, 31(8), 3127–3141.
10. Wireshark Foundation. (2023). *Wireshark User's Guide*.  
<https://www.wireshark.org>
11. Shafiq, M. Z., Yu, X., Wang, A. X., Laghari, A. A., & Khokhar, A. (2016). *Network traffic anomaly detection using machine learning techniques*. Journal of Network and Computer Applications, 85, 29–43.

