

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 akışta 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 sınıfa ayrılır;

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

- 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 trafiği çoğu zaman normalden farklı paket boyutu/hacim davranış 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

- $\text{syn_fin_ratio} = \text{syn_count} / (\text{fin_count} + 1)$

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 ($\text{select_dtypes}(\text{include}=[\text{np.number}])$)
- inf / $-\text{inf}$ 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 $\text{MinMaxScaler}(\text{feature_range}=(0,1))$ 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)
- $\text{hidden_dim}=64$, $\text{num_layers}=2$, $\text{dropout}=0.3$
- $\text{batch_first}=\text{True}$: veri "batch" odaklı gelir.

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

2) Çıkış Katmanı

Son zaman adımının hidden state'i 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 sızıntısını (data leakage) önlemek için kritiktir.

3) Sequence (Zaman Penceresi) Oluşturma Mantığı

Bu projede kritik tasarım: model tek satıra değil, ardışık 10 satıra 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 saldırı varsa o sequence “Attack” sayılır.

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

VII. Eğitim Süreci

1) Veri Bölme:

Önce tüm normal + saldırı 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 çıktısı olasılığa çevrilir ve `threshold=0.5` ile sınıf kararı üretilir.

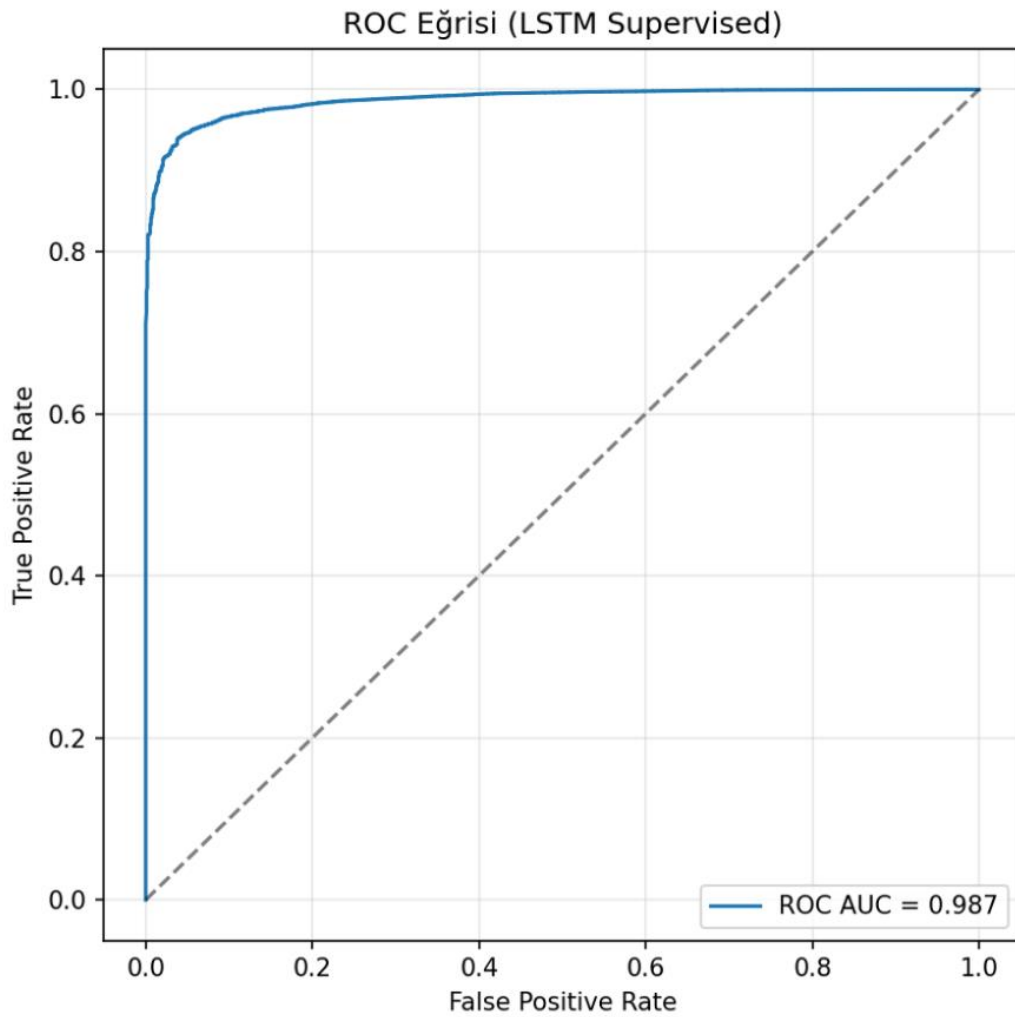
Confusion Matrix ve Classification Report

Confusion matrix ile:

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

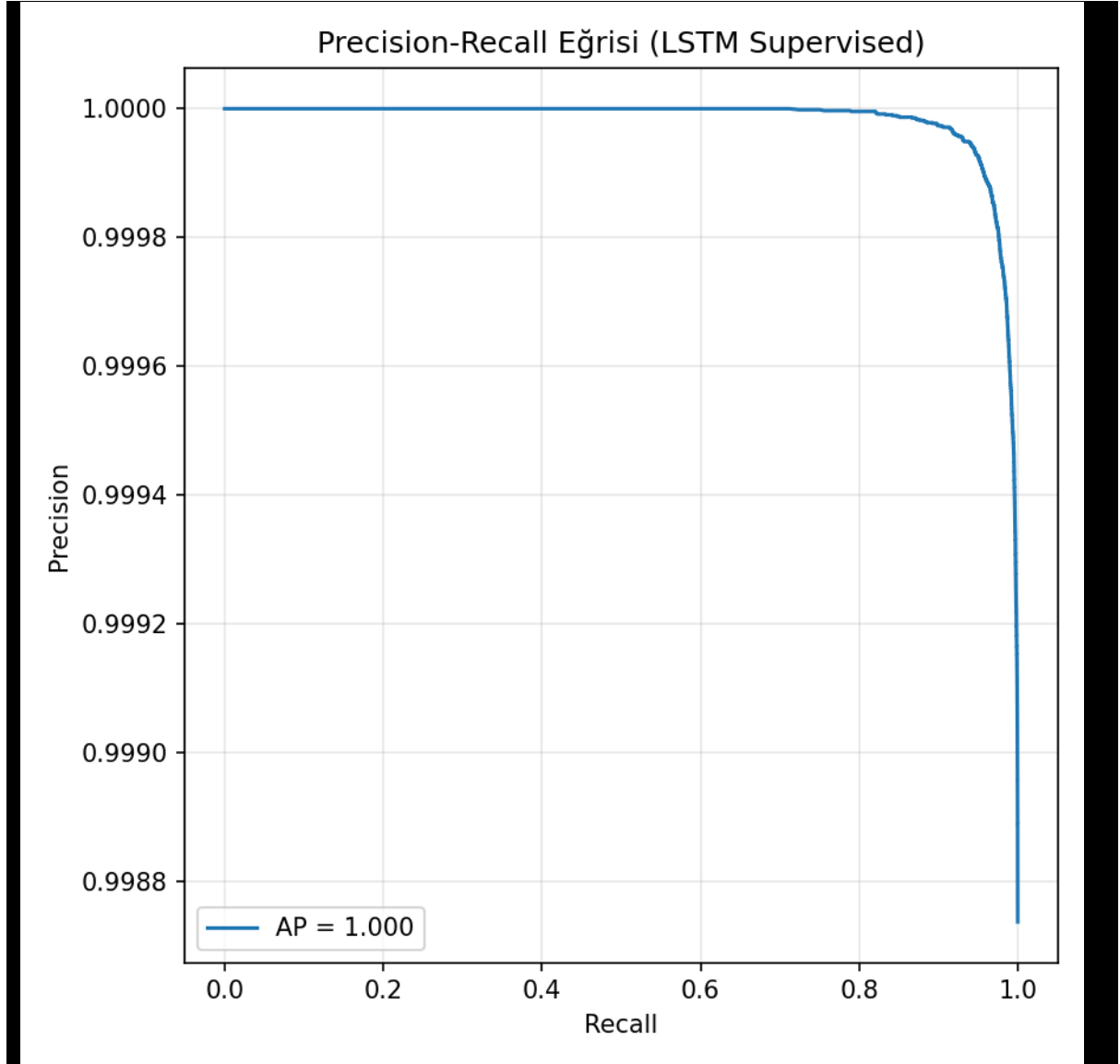
Classification report; precision/recall/f1 değerlerini sınıf bazında verir. Bu metrikler, sistemin “kaç saldırı yakaladığı” ve “kaç yanlış alarm verdiği” gibi operasyonel açıdan önemli noktaları gösterir.

ROC Eğrisi ve AUC:



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

Precision–Recall Eğrisi ve AP:

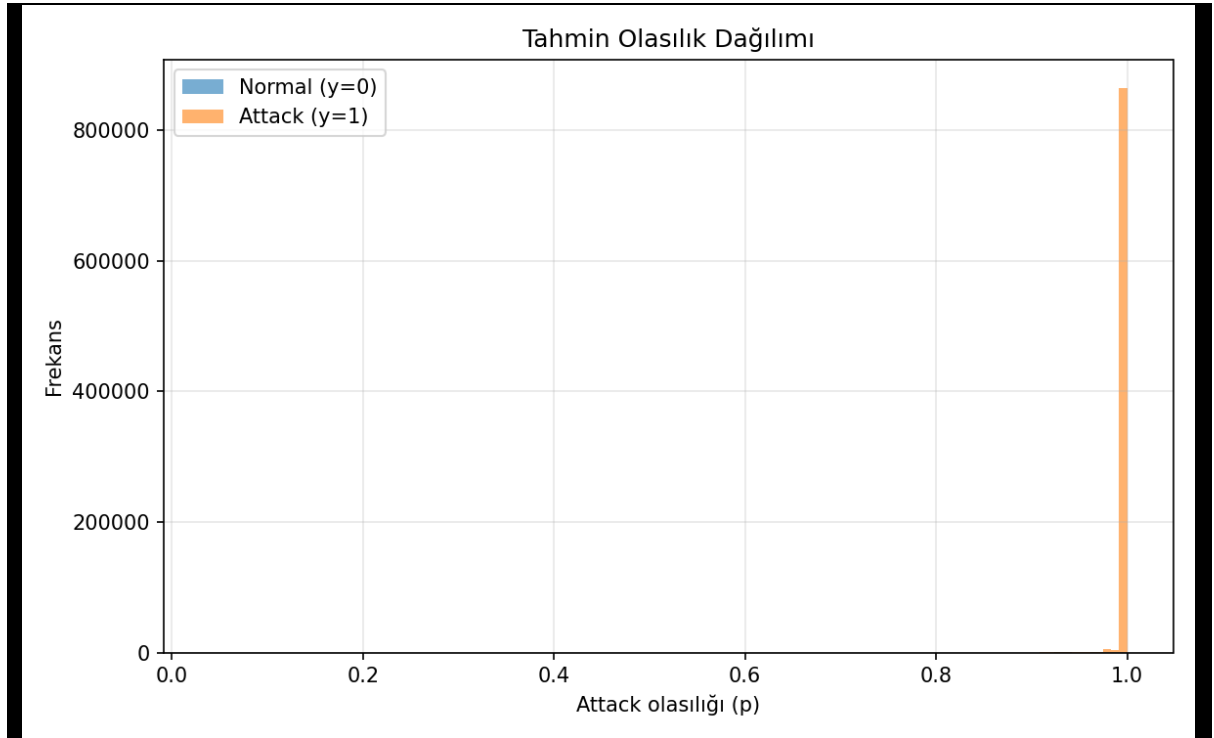


- **AP = 1.000**

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 ayırımında sızıntı (data leakage) ihtimali gibi durumlarda da görülebilir. Bu nedenle veri ayırım 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. Gerçek Zamanlı Anomali Tespiti

1) Buffer Mekanizması

Detect_anomaly() fonksiyonu gelen tek satırı:

1. Temizler
2. scaler 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 ağın 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.

