BURSA TEKNİK ÜNİVERSİTESİ

Mühendislik ve Doğa Bilimleri Fakültesi – Bilgisayar Mühendisliği Bölümü

BLM0326 Bilgisayar Ağları

Bahar 2025

Advanced Secure File Transfer System with Low-Level IP Processing & Network Performance Analysis - The Final Report

Esra İLBOĞA

21360859063

İçindekiler

1. GİRİŞ

Proje kapsamında, hem **UDP (User Datagram Protocol)** hem de **TCP (Transmission Control Protocol)** üzerinden dosya aktarımı desteklenmiştir. Dosyalar, aktarım öncesinde **AES-256 (Advanced Encryption Standard)** algoritması kullanılarak şifrelenmekte ve AES anahtarı **RSA-2048 (Rivest-Shamir-Adleman)** algoritmasıyla şifrelenerek güvenli bir şekilde iletilmektedir. Böylece, dosyanın yalnızca doğru anahtara sahip alıcı tarafından çözülebilmesi sağlanmıştır.

Ayrıca dosya bütünlüğü, **SHA-256 hash algoritması** kullanılarak her bir parça için ayrı ayrı kontrol edilmektedir. Bu sayede, özellikle UDP gibi güvenilir olmayan protokollerde parça kaybı veya bozulmalar tespit edilebilmekte ve güvenli veri iletimi garanti altına alınmaktadır.

Sistem sadece arka planda çalışan bir yazılım değil, aynı zamanda **grafiksel kullanıcı arayüzü (GUI)** içeren etkileşimli bir uygulama olarak tasarlanmıştır. Kullanıcılar, dosya seçimi, protokol seçimi (TCP/UDP), token ile doğrulama ve gönderim işlemlerini kolayca gerçekleştirebilmektedir.

Projenin ileri seviye bileşenleri şunlardır:

- Paket kaybı simülasyonu içerir (hem kodla hem Clumsy aracıyla).
- MITM saldırısı simülasyonu yapılabilir.
- iPerf3 ve Pingile performans analizi gerçekleştirir.
- Wireshark ile ağ trafiği izlenir ve analiz edilir.

2. SİSTEM MİMARİSİ VE GENEL YAPI

Projenin sistem mimarisi istemci-sunucu modeli üzerine kurulmuştur ve modüler yapıda geliştirilmiştir. Tüm bileşenler; güvenli veri aktarımı, kullanıcı etkileşimi, performans ölçümü ve güvenlik analizi gibi farklı işlevleri yerine getirecek şekilde ayrılmıştır. Grafiksel kullanıcı arayüzü (GUI) üzerinden kullanıcı, seçtiği protokol (TCP/UDP) doğrultusunda dosya gönderim sürecini başlatırken, arka planda şifreleme, bütünlük kontrolü, ağ analizi ve saldırı simülasyonu gibi işlemler eş zamanlı olarak yürütülür. Bu yapı, hem teknik bütünlüğü hem de kullanıcı deneyimini ön planda tutan bir sistem sunar.

Bileşen	Açıklama
Client (İstemci)	Dosya seçer, şifreler ve gönderir.
Server (Sunucu)	Protokole göre dinler, dosyayı çözer ve kaydeder.
GUI	Protokol seçimi, token girişi, dosya gönderme butonu.
Şifreleme	AES-256 + RSA-2048 kullanılır.
Performans Ölçüm	iPerf3 + ping + tabulate ile ölçüm ve tablo oluşturma.
Saldırı Simülasyonu	MITM ve paket kaybı simülasyonu.

3. TEKNİK DETAYLAR

3.1. Şifreleme ve Kimlik Doğrulama

- Dosyalar önce **AES-256 (EAX)** ile şifrelenir.
- AES anahtarı, alıcının **RSA-2048** açık anahtarı ile şifrelenir.
- Gönderim sırasında token değeri zorunlu tutulur
- Dosya bütünlüğü için SHA-256 ile her parça ayrı kontrol edilir.

```python
def aes_encrypt(data, key):
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(data)
    return cipher.nonce, ciphertext, tag
```

Görsel 3.1.1 AES.MODE_EAX modunda şifreleme yaparak veriyi AES-256 ile şifreler

```python
def rsa_encrypt(key, public_key):
    cipher_rsa = PKCS1_OAEP.new(public_key)
    return cipher_rsa.encrypt(key)
```

Görsel 3.1.2 AES anahtarı, burada RSA algoritmasıyla şifrelenir.

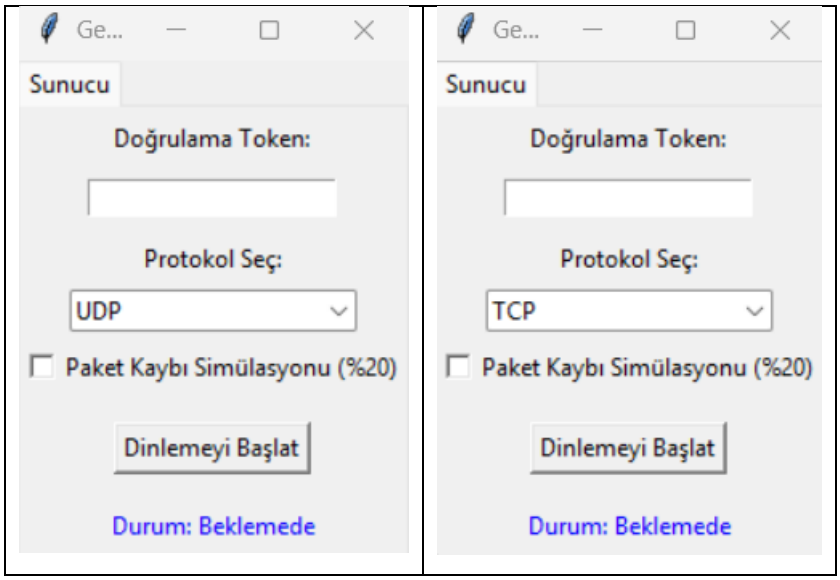
```python
packet = pickle.dumps({
    'aes_key': encrypted_key,
    'nonce': nonce,
    'tag': tag,
    'data': encrypted,
    'token': self.token.get(),
    'filename': os.path.basename(dosya)
})
```

Görsel 3.1.3 client_gui.py dosyasında yer alır. token bu şekilde paket içine gömülür ve sunucu tarafında kontrol edilir.

3.2. UDP & TCP Destekli Aktarım

UDP ve TCP protokolleri, dosya iletimine farklı yaklaşımlar sunar. UDP, dosyayı parçalara ayırarak SHA-256 ile bütünlük kontrolü yapar; eksik veya bozuk parça varsa dosya çözülmez. TCP ise tek parça iletim ve dahili hata kontrolü sağlar ancak daha yavaştır. UDP tarafında %20'ye kadar paket kaybı simülasyonu GUI üzerinden etkinleştirilebilir. TCP, kendi tekrar mekanizması sayesinde buna ihtiyaç duymaz. Her iki protokolde de protokol seçimi, token doğrulama ve bütünlük kontrolü, kullanıcı dostu arayüz ile sağlanır.

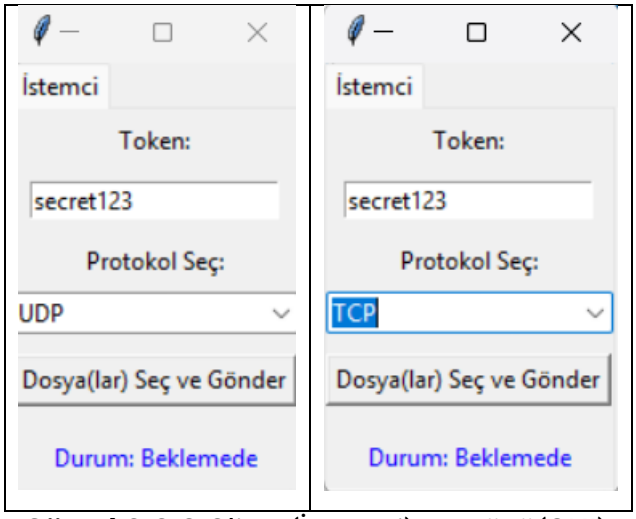
Özellik	**UDP**	**TCP**
Parçalara Bölme	Evet	Hayır
Bütünlük Kontrolü	SHA-256 ile	Kısmi (eksik çözülürse hata)
Paket Kaybı	Kod içi simülasyon & Clumsy	Hayır
Performans	Daha hızlı	Daha güvenli



Görsel 3.2.1 Sunucu arayüzü(GUI)

Görsel 3.2.1'de görülen kullanıcı arayüzü (GUI), sunucu tarafında dosya alım sürecini yönetmek üzere geliştirilmiştir. Kullanıcı, öncelikle doğrulama token'ını girerek güvenliği sağlar; ardından UDP veya TCP protokolünü seçebilir. UDP için ek olarak %20 oranında paket kaybı simülasyonu etkinleştirilebilir. "Dinlemeyi Başlat" butonuyla süreç başlatılır ve altta yer alan durum etiketi, bağlantının

güncel durumunu kullanıcıya iletir. Bu yapı, hem işlevsel testler hem de güvenlik analizleri için kullanıcı dostu ve gelişmiş bir kontrol paneli sunar.



Görsel 3.2.2 Client(İstemci) arayüzü(GUI)

Görsel 3.2.2'de gösterilen GUI istemci tarafında dosya gönderimini yönetmek için tasarlanmıştır. Kullanıcı, önce güvenlik için token bilgisini girer; ardından UDP veya TCP protokolünü seçebilir. "Dosya(lar) Seç ve Gönder" butonuyla birden fazla dosya seçilerek güvenli biçimde sunucuya iletilir. Arayüz, işlem sürecini alt kısımda "Durum" etiketi ile anlık olarak kullanıcıya bildirir. Hem UDP hem TCP desteği sayesinde esnek ve güvenli bir dosya aktarımı sunar.

3.3. IP Katmanı Manipülasyonu

paketleme.py ile **Scapy** kullanılarak:

- TTL değeri değiştirildi.
- Fragmentation flag ayarlandı (DF set edildi).
- Raw payload olarak ASCII veri eklendi.
- Source IP spoofing uygulandı (MITM simülasyonu için)

```python
from scapy.all import IP, send, Raw

def gonder_ozel_paket(data):
    ip_katmani = IP(dst="127.0.0.1", ttl=64, flags=2)  # DF bayrağı
    paket = ip_katmani / Raw(load=data)
    send(paket)
```

Görsel 3.3.1 paketlem.py

Görsel 3.3.1'de **Scapy** ile IP seviyesinde özel bir paket oluşturulur ve gönderilir:

- **TTL=64** ile zaman aşımı değeri,

- **flags=2** ile **DF (Don't Fragment)** bayrağı ayarlanır,
- **Raw payload** kısmına ASCII veri eklenir.

Amaç, IP katmanında paket manipülasyonu yaparak ağ davranışlarını test etmektir.

3.4. MITM Saldırı Simülasyonu

```python
from scapy.all import *

def mitm_simulasyonu():
    print("Sahte paket gönderiliyor...")
    sahte_paket = IP(src="192.168.1.1", dst="127.0.0.1")/UDP(sport=1234, dport=5005)/Raw(load=b"MITM Attack")
    send(sahte_paket)
```

Görsel 3.4.1 mitm_simulasyon.py mitm_simülasyonu()

Görsel 3.3.1 ile **Scapy** kütüphanesiyle sahte bir UDP paketi oluşturup gönderir.

- Kaynak IP sahte olarak "192.168.1.1" ayarlanmıştır (spoofing).
- Hedef "127.0.0.1"'dir (localhost).
- Veri kısmına "MITM Attack" yazısı yüklenmiştir.

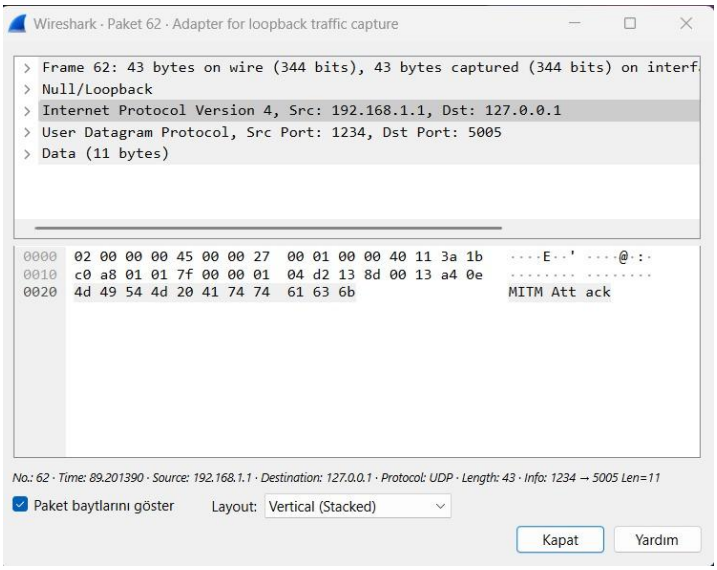
Amaç, **MITM (Ortadaki Adam) saldırısını** simüle ederek sistemin tepkisini test etmektir.

```
PS C:\Users\ilbog\OneDrive\Masaüstü\secure_file_transfer> python
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from mitm_simulasyon import mitm_simulasyonu
>>> mitm_simulasyonu()
Sahte paket gönderiliyor...
.
Sent 1 packets.
```

Görsel 3.4.2 mitm_simülasyonu()

Görsel 3.4.2'deki kod ile mitm_simulasyonu() fonksiyonu çağrılarak sahte bir UDP paketi, spoof edilmiş (sahte) kaynak IP ile gönderilmiştir. Bu işlem, bir MITM (Man-in-the-Middle) saldırısı simülasyonu amacıyla yapılır.

mitm_simulasyonu() fonksiyonu ile gönderilen **sahte paketin** detayları (kaynak IP, hedef IP, port, payload içeriği) Wireshark ile doğrulanmıştır. İlgili Wireshark çıktısı **Görsel 3.4.3**'te gösterilmiştir.



Görsel 3.4.3 mitm simülasyonu Wireshark çıktısı

Alan	Değer
Source	192.168.1.1
Destination	127.0.0.1
Protocol	UDP
Payload	"MITM Attack"

Tablo 3.4.1 Mitm saldırısı sonucu Wireshark'tan elde edilen bilgiler

3.5. Paket Kaybı Simülasyonu

1. Kodla paket kaybı simülasyonu gerçekleştirilir. network_utils.py dosyası içinde **UDP paketlerinin belirli bir oranda (%20)** rastgele atlanmasını sağlayan kod bloğu bulunmaktadır.

```
def receive_packets(timeout=8, simulate_loss=False, loss_rate=0.2):
```

Görsel 3.5.1 network_utils.py recieve_packets fonksiyonu

Görsel 3.2.1'de görüleceği üzere sunucu GUI ekranında **paket kaybı simülasyonu(%20)** kutucuğu işaretlendiğinde paket kaybı simülasyonu için server_gui.py dosyası içindeki ilgili kodlar ile paket kaybı simülasyonu gerçekleştirilir.

```
# ✅ Simülasyon parametresi gönderildi
packets = receive_packets(simulate_loss=self.simulate_loss.get(), loss_rate=0.2)
```

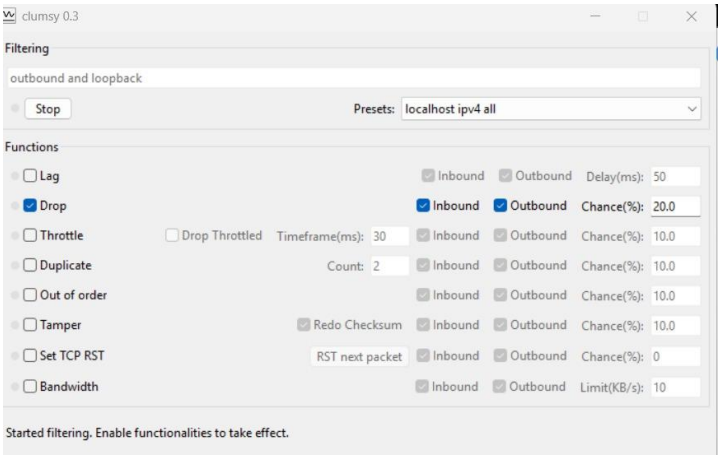
Görsel 3.5.2 server_gui.py içindeki simülasyon parametre gönderimi

İlgili kutucuk seçilip çıktı alındığından **Görsel 3.5.3**'teki gibi görünür.

```
UDP] Dinleme başladı... Simülasyon: AÇIK
Simülasyon] Paket 3 atlandı (kayıp).
Simülasyon] Paket 5 atlandı (kayıp).
Simülasyon] Paket 7 atlandı (kayıp).
Simülasyon] Paket 8 atlandı (kayıp).
Simülasyon] Paket 15 atlandı (kayıp).
Simülasyon] Paket 33 atlandı (kayıp).
Simülasyon] Paket 34 atlandı (kayıp).
Simülasyon] Paket 36 atlandı (kayıp).
Simülasyon] Paket 46 atlandı (kayıp).
Simülasyon] Paket 49 atlandı (kayıp).
Simülasyon] Paket 52 atlandı (kayıp).
UDP] Son paket alındı.
Uyarı] Eksik parçalar: [3, 5, 7, 8, 15, 33, 34, 36, 46, 49]
```

Görsel 3.5.3 kod ile paket kaybı simülasyonu çıktısı

2. Clumsy aracıyla da paket kaybı simüle edilebilir.
    - Clumsy.exe açılır
    - Drop seçilir
    - %20 yazılır
    - Port: 5005 → Start



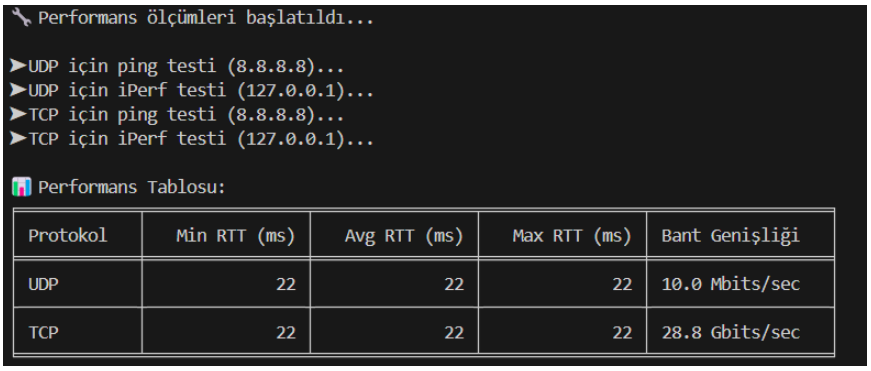
Görsel 3.5.4 clumsy kullanarak paket kaybı simülasyonu

4. PERFORMANS ANALİZİ

Sistemin ağ performansını analiz etmek için geliştirilmiş bir test aracıdır.
Kullanılan araçlar:

- iperf3: TCP & UDP bant genişliği,
- ping: Gecikme ölçümü,
- tabulate: Tablo oluşturma,
- Çıktılar performans_raporu.txt dosyasına kaydedilir.

ipref çıktısı alabilmek için iperf3.exe -s komutu başka bir terminalde çalışıyor olmalıdır.



Protokol	Min RTT (ms)	Avg RTT (ms)	Max RTT (ms)	Bant Genişliği
UDP	22	22	22	10.0 Mbits/sec
TCP	22	22	22	28.8 Gbits/sec

Görsel 4.1 performans_analizi.py çıktısı

- **Ping testi (8.8.8.8):**
  Google DNS adresine 4 kez ping atarak **gecikme sürelerini (min, avg, max RTT)** ölçer.
- **iPerf3 testi (127.0.0.1):**
  Lokal iperf3 sunucusuna TCP ve UDP üzerinden bağlantı kurarak **bant genişliğini (throughput)** ölçer.
- **Sonuçların analiz edilmesi:**
  RTT değerleri ping çıktısından, bant genişliği ise iperf3 çıktısından ayrıştırılır.
- **Tablo halinde çıktı:**
  Elde edilen metrikler tabulate modülüyle bir tabloya dönüştürülür ve hem ekrana yazdırılır hem de **performans_raporu.txt** dosyasına kaydedilir.

Bu sayede kullanıcı, sistemin UDP ve TCP protokolleri altındaki **iletişim kalitesini ölçüp kıyaslayabilir.**

5. SONUÇ

Bu proje kapsamında, güvenli ve performans odaklı bir dosya aktarım sistemi başarıyla geliştirilmiştir. Sistem, hem TCP hem de UDP protokolleri üzerinde çalışabilmekte, her iki protokol için özel avantajlar sunmaktadır. Tüm dosya iletimleri AES-256 şifreleme algoritması ile güvence altına alınmış, anahtar yönetimi ise RSA-2048 ile sağlanmıştır. Ayrıca, her veri parçasının bütünlüğü SHA-256 özetiyle kontrol edilerek, güvenliğin sadece gizlilik değil doğruluk boyutunda da sağlandığı bir yapı kurulmuştur.

Sisteme entegre edilen **token tabanlı doğrulama**, istemci ve sunucu arasında yalnızca yetkili kullanıcıların iletişim kurmasını garanti altına alırken, GUI üzerinden kullanıcıya

protokol seçimi, dosya gönderimi ve simülasyon özellikleri gibi birçok kontrol kolaylığı sunulmuştur.

Geliştirilen sistem, aynı zamanda:

- **IP katmanı üzerinde TTL ve fragmentasyon bayrağı gibi alanları değiştirme** (Scapy ile),
- **MITM saldırısı simülasyonu** (sahte kaynak IP ile paket enjeksiyonu),
- **Paket kaybı testleri** (hem kod düzeyinde hem de Clumsy gibi harici araçlarla),
- **iPerf3 ve ping ölçümleri ile bant genişliği ve gecikme analizi** (tabulate ile görselleştirilmiş),
- **UDP tarafında kayıp paket simülasyonuna bağlı olarak dosya çözüm reddi**

gibi ileri seviye analiz ve güvenlik testleriyle desteklenmiştir.

Sonuç olarak, sistem sadece temel veri iletimi değil, aynı zamanda güvenlik, ağ davranışları ve performans ölçümü gibi çok boyutlu ihtiyaçlara karşılık verebilecek şekilde tasarlanmıştır. Proje, hem akademik öğrenme süreci hem de pratik ağ mühendisliği açısından yüksek katkı sağlamıştır. Gelecekte, sistemin çoklu bağlantı desteği, log mekanizması, oturum tabanlı anahtar yönetimi ve web/mobil GUI entegrasyonlarıyla daha da geliştirilebilir hale gelmesi mümkündür.

6. EKSİKLER VE GELİŞTİRME ALANLARI

TCP ve UDP üzerinden güvenli dosya aktarımı, kullanıcı dostu arayüz, şifreleme ve performans analizi gibi birçok temel özelliği başarıyla içermektedir. Ancak aşağıdaki noktalar, gelecekte sistemin daha da iyileştirilmesine olanak tanıyacak eksikler ve önerilerdir:

- **Çoklu Dosya TCP Aktarımı:**
  TCP tarafında her bir bağlantı için ayrı işlem başlatılması gerektiğinden çoklu gönderim performans ve doğruluk açısından sınırlandırılmıştır. Çoklu TCP gönderimi için her dosya için yeni bağlantı oluşturma veya birden fazla dosyayı birlikte seri hâlde işleme desteği eklenebilir.
- **Çoklu Dosya UDP Aktarımı:**

UDP ile dosya aktarımı çoklu yapılmaya çalışıldığı halde bir dosya dışındakileri hata vererek aktarıma müsaade etmemiştir. Bu durumda çoklu dosya gönderimi UDP için geliştirilebilir.

- **Paket Sıralama ve Yeniden Gönderim (UDP):**
  UDP'de parçalar doğru sırada alınmakta, ancak kayıp paketlerde yalnızca uyarı verilmekte ve çözümleme yapılmamaktadır. Daha sağlam bir yapı için **paket tekrar gönderimi (retransmission)** veya eksik paket bildirimi protokolü geliştirilebilir.
- **Dosya Türü Tespiti ve MIME Desteği:**
  Şu an tüm dosyalar ikili (binary) olarak ele alınıyor. Sistem, dosya uzantılarına göre MIME türü belirleyerek örneğin .jpg dosyasının açılıp açılmadığını test edebilir, veya GUI'de "desteklenen türler" filtresi sunulabilir.
- **Performans Ölçümü Gerçek Trafikte Yapılamıyor:**
  iPerf testleri localhost üzerinden yapılmakta ve ping verisi dış IP (8.8.8.8) ile alınmaktadır. Gerçek ağlar arası veri trafiği simülasyonu, sanal makineler veya iki ayrı fiziksel cihaz arasında denenerek daha doğru bant genişliği ölçümleri yapılabilir.
- **Gelişmiş Loglama ve Hata Kaydı:**
  Anlık hatalar sadece messagebox uyarıları ile gösterilmektedir. Kalıcı bir hata günlüğü (log dosyası) oluşturarak kullanıcıya rapor sunulabilir.
- **Şifreleme Anahtar Yönetimi:**
  RSA anahtarları dosya olarak projede yer almakta ve sabittir. Daha güçlü bir sistemde **anahtar yenileme mekanizması**, oturum bazlı anahtar üretimi veya kullanıcıya özel anahtar dizisi tanımlanabilir.
- **MITM Simülasyonu:**
  Scapy ile temel sahte paket gönderimi yapılmıştır. Ancak sahte paketlerin kullanıcı oturumlarını taklit etmesi veya geçerli token ile gelmesi gibi gelişmiş senaryolar eklenerek daha derinlemesine bir MITM analizi yapılabilir.
- **Mobil veya Web Arayüz Desteği:**
  GUI yalnızca masaüstü ortamında çalışmaktadır. İleri düzey kullanıcı erişimi için web tabanlı veya mobil uyumlu bir arayüz geliştirilebilir.

7. YOUTUBE VE GİTHUB

- Github Linki :
  https://github.com/Esra-ilboga/Secure-File-Transfer-System.git
- Youtube Linki :
  https://youtu.be/fG-8mG5yuCE

8. KAYNAKLAR

- Wireshark Foundation. (2024). Wireshark User Guide.
  Wireshark User's Guide/
- Scapy – Packet Manipulation Tool.
  https://scapy.net
- iPerf3 – Network Bandwidth Measurement Tool.
  https://github.com/esnet/iperf
- Python Software Foundation.
  https://docs.python.org/3/
- Tabulate Library – Python Pretty Tables.
  https://pypi.org/project/tabulate/
- Tkinter – GUI Programming in Python.
  https://docs.python.org/3/library/tkinter.html
- Clumsy – Network Traffic Simulator.
  https://jagt.github.io/clumsy/