ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM4510 – IOS ile Mobil Uygulama Geliştirme

Port Tarayıcı Projesi

Abdullah Gedik

20290251

İçindekiler

---

1. Giriş

Bu Python tabanlı port tarayıcı, bir hedef IP adresi veya alan adı üzerindeki TCP portlarının “AÇIK” ya da “KAPALI/Filtrelenmiş” olduğunu hızlıca raporlar. Araç; socket ve threading gibi standart kütüphaneler kullanılarak, komut satırı (CLI) üzerinden çalışır ve port tarama sonuçlarını ekranda listeler.

Amaç ve Kullanım Alanları:

- **Ağ Keşfi & Güvenlik:** Hangi servislerin çalıştığını tespit ederek gereksiz açık portları kapatma ve firewall yapılandırmalarını doğrulama imkânı.
- **Otomasyon & Test:** CLI komutlarıyla betiklere eklenebilir, periyodik tarama ve raporlama süreçlerine entegre edilebilir.
- **Eğitim & Geliştirme:** Socket programlama ve çoklu iş parçacığı kullanımını öğrenme veya doğrulama amacıyla pratik bir laboratuvar aracı olarak hizmet eder.

Kapsam ve Sınırlamalar:

- Yalnızca TCP portlarına full-open (connect_ex) tarama.
- IPv4 adres desteği; UDP, IPv6 ve ileri teknikler (banner grabbing, OS algılama) proje dışında.
- CLI tabanlı çıktı; grafik veya dosya bazlı raporlama eklenmemiştir.

2. Temel Kavramlar ve Literatür Özeti

2.1 Port ve Servis

- **Port:** 0–65535 arası numaralanan, verinin doğru uygulamaya iletilmesini sağlayan mantıksal adres.
- **Well-Known Ports (0–1023):** HTTP (80), HTTPS (443), SSH (22) gibi yaygın servisler.
- **Servis:** Port üzerinde dinleyerek gelen bağlantıları işleyen sunucu yazılımı.

2.2 TCP/IP'de Bağlantı Modeli

- **TCP:** Üç aşamalı handshake ile güvenilir veri aktarımı sağlar.
- **Socket Programlama:** Python'da socket.socket(), connect(), send() ve recv() ile ağ iletişimi yapılır.
- **Zaman Aşımı:** settimeout() sayesinde yavaş cevap veren veya kapalı portlarda bekleme süresi sınırlandırılır.

2.3 Port Tarayıcılar

- **Nmap:** SYN-scan, UDP tarama, banner grabbing ve OS algılama gibi ileri özelliklerin bulunduğu endüstri standardı araç.
- **Basit Scanner'lar:** Full-open yöntemle çalışan, genellikle tek iş parçacıklı ve öğrenme odaklı araçlar (Thomas D. Nadeau, *Practical Network Programming with Python*; Gordon Lyon, *Nmap Network Scanning*).

3. Proje Tasarımı ve Yöntem

3.1 Kullanılan Araçlar ve Kütüphaneler

- **Python 3.x**
- **socket**: TCP bağlantı denemeleri için.
- **threading**: Eşzamanlı tarama.
- **argparse**: CLI argümanları.
- **queue**: İş parçacıkları arası görev paylaşımı.
-

3.2 Mimari ve Akış Diyagramı

1. CLI'den hedef, port aralığı ve zaman aşımı parametreleri okunur.
2. Port numaraları bir iş kuyruğuna (queue.Queue) eklenir.
3. Belirlenen sayıda iş parçacığı başlatılır, her biri kuyruğu tüketerek connect_ex ile tarama yapar.
4. Sonuçlar thread-safe bir listeye kaydedilir ve tarama tamamlandığında ekrana yazdırılır.
5.

3.3 Tarama Algoritması (Pseudocode)

```
function scan_port(target, port, timeout):
  sock = socket(AF_INET, SOCK_STREAM)
  sock.settimeout(timeout)
  result = sock.connect_ex((target, port))
  return "OPEN" if result == 0 else "CLOSED"

main():
  args = parse_cli_args()
  fill work_queue with range(start_port, end_port)
  spawn N threads running worker()
  wait for queue.join()
  print sorted(results)
```

4. Uygulama Detayları

4.1 Proje Dosya Yapısı

port_scanner/

├── cli.py

├── scanner.py

├── worker.py

├── utils.py

└── requirements.txt

4.2 Örnek Kod Parçacıkları

- **cli.py**: argparse ile parametre işleme.
- **scanner.py**: PortScanner sınıfı, thread pool yönetimi.
- **worker.py**: connect_ex tabanlı port tarama.
- **utils.py**: Hedef doğrulama.

4.3 Kullanım

python cli.py -t localhost -p 1-100 -to 0.5 -th 10

5. Test ve Sonuçlar

5.1 Deneme Ortamı

- **İşletim Sistemi:** Windows
- **Python 3.12.2**
- **Test Servisleri:** SSH(22), HTTP(80), Custom(5023)

5.2 Örnek Sonuç

Port 20–25 (–th 10, –to 0.5)

Port 22: OPEN

Diğerleri: CLOSED

Performans (1–100 arası):

- Tek-thread: ~12.8s
- 10-thread: ~2.3s

---

6. Tartışma ve Geliştirme Önerileri

- **Mevcut Kısıtlar:** Full-open scan, IPv4, CLI tabanlı.
- **Öneriler:** UDP tarama, banner grabbing, SYN-scan, asyncio, GUI, dosya/JSON/CSV çıktı

---

7. Sonuç

Bu proje, Python standard kütüphaneleriyle temel TCP port tarayıcı geliştirme, socket ve threading kavramlarını pekiştirme imkânı sundu. Performans testleri, thread sayısının tarama süresini ciddi ölçüde kısalttığını gösterdi. İlerleyen aşamalarda ileri tarama yöntemleri ve kullanıcı dostu arayüzlerle proje zenginleştirilebilir.

---

8. Kaynakça

1. Python Software Foundation. “socket — Low-level networking interface.” Python 3.9 Documentation.
2. Python Software Foundation. “threading — Thread-based parallelism.” Python 3.9 Documentation.
3. Lyon, Gordon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.Com LLC, 2009.
4. Nadeau, Thomas D. *Practical Network Programming with Python*. Apress, 2015.