KAPADOKYA ÜNİVERSİTESİ

BİLGİSAYAR PROGRAMCILIĞI 1 SINIF

PYTHON II – VİZE ÖDEVİ

KAPADOKA SİBER TEKNİK GRUBU

Kod betiği Örneği:

import socket

import sys

import os

import threading

from colorama import Fore, Style

**if** len(sys.argv) != 3:

print("Kullanım <host> <port\_aralığı> şeklinde.")

sys.exit(1)

host = sys.argv[1]

port\_range = sys.argv[2]

low\_port = int(port\_range.split("-")[0])

high\_port = int(port\_range.split("-")[1])

banner = r"""

█████ ████ █████████ ███ ███████████

░░███ ███░ ███░░░░░███ ░░░ ░█░░░███░░░█

░███ ███ ██████ ████████ ██████ ░███ ░░░ ████ ░ ░███ ░ ██████

░███████ ░░░░░███ ░░███░░███ ░░░░░███ ░░█████████ ░░███ ░███ ███░░███

░███░░███ ███████ ░███ ░███ ███████ ░░░░░░░░███ ░███ ░███ ░███████

░███ ░░███ ███░░███ ░███ ░███ ███░░███ ███ ░███ ░███ ░███ ░███░░░

█████ ░░████░░████████ ░███████ ░░████████░░█████████ █████ █████ ░░██████

░░░░░ ░░░░ ░░░░░░░░ ░███░░░ ░░░░░░░░ ░░░░░░░░░ ░░░░░ ░░░░░ ░░░░░░

░███

█████

░░░░░ KAPADOKYA ÜNİVERSİTESİ - Siber Teknik Grubu 2023

"""

# Bannerinizin renklerini belirleyin

colors = [Fore.RED, Fore.YELLOW, Fore.GREEN, Fore.CYAN, Fore.BLUE, Fore.MAGENTA]

# Bannerinizin her satırını renklendirin

colored\_banner = ""

**for** i, line **in** enumerate(banner.split("\n")):

colored\_banner += colors[i % len(colors)] + line + "\n"

print(colored\_banner + Style.RESET\_ALL)

open\_ports = []

closed\_ports = 0

lock = threading.Lock()

def scan\_port(port):

global closed\_ports

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.settimeout(0.5)

result = s.connect\_ex((host, port))

**if** result == 0:

**with** lock:

print(Fore.GREEN + f"Port {port}: Açık" + Style.RESET\_ALL)

open\_ports.append(port)

**else**:

**with** lock:

closed\_ports += 1

s.close()

threads = []

**for** port **in** range(low\_port, high\_port + 1):

t = threading.Thread(target=scan\_port, args=(port,))

t.start()

threads.append(t)

**for** t **in** threads:

t.join()

print(Fore.RED + f"Taranan Port: {closed\_ports}")

print(Style.RESET\_ALL)

**if** open\_ports:

print(f"Açık portlar Nmap'e gönderiliyor:")

ports\_str = ",".join(str(p) **for** p **in** open\_ports)

os.system(f"nmap -p {ports\_str} -sV {host}")

**else**:

print("açık yada böyle bir host yok.")

**Çalışma mantğı:**

1. Betik, komut satırından alınan argümanları kullanarak hedef **host** ve **port\_range**'i alır.
2. **port\_range** stringini '-' karakterine göre böler ve **low\_port** ve **high\_port** değerlerini belirler.
3. Terminalde renkli bir banner gösterir. Bu sadece görsel amaçlıdır ve betiğin işlevselliği ile ilgisi yoktur.
4. **open\_ports** adlı boş bir liste oluşturur ve **closed\_ports** adlı bir sayaç başlatır. Bu iki değişken, betik çalışırken taranan portların sonuçlarını saklamak için kullanılır.
5. **scan\_port** adlı bir fonksiyon tanımlar. Bu fonksiyon, belirtilen **host** ve **port** için bağlantıyı deneyerek portun açık olup olmadığını kontrol eder. Bağlantı başarılıysa, port açıktır ve **open\_ports** listesine eklenir. Başarısız olursa, port kapalıdır ve **closed\_ports** sayacı artırılır.
6. Tarama işlemini hızlandırmak için, her port için **scan\_port** fonksiyonunu çalıştıran bir dizi iş parçacığı oluşturur.
7. Tüm iş parçacıklarının tamamlanmasını bekler.
8. Taradığı toplam kapalı port sayısını ekrana yazdırır.
9. Eğer açık portlar bulunmuşsa, bunları Nmap'e göndererek daha fazla bilgi elde etmek için kullanır.
10. Eğer açık port bulunamazsa, "açık yada böyle bir host yok" mesajını ekrana yazdırır.

Kısacası, bu betik belirtilen bir **host** ve **port\_range** için açık portları tespit etmeye çalışır ve açık portları Nmap ile daha fazla incelemek için kullanır. Betik, iş parçacığı kullanarak tarama işlemini hızlandırır ve colorama kütüphanesini kullanarak terminal çıktılarını renklendirir.

**Teknik Açıklama:**

Bu betik bir **sys** kütüphanesi içindeki **argv** fonksiyonunu kullanarak kullanıcıdan iki argüman alır. Bu iki argüman, **host** adında bir değişkene ve **port\_range** adında bir başka değişkene atanır.

İlk if koşulu, **len** fonksiyonunu kullanarak **sys.argv** listesinde kaç öğe olduğunu kontrol eder. Eğer liste uzunluğu 3 değilse (yani kullanıcı **host** ve **port\_range**'i girmediyse) o zaman "Kullanım <host> <port\_aralığı> şeklinde." şeklinde bir hata mesajı verir ve programı sonlandırır (**sys.exit(1)**).

**host** değişkeni, kullanıcının girdiği ilk argüman olan IP adresini tutar.

**port\_range** değişkeni, kullanıcının girdiği ikinci argüman olan port aralığını tutar. Bu argüman, başlangıç ve bitiş port numaraları arasında bir tire ile ayrılmış iki sayıdır. Örneğin, "100-200" bir port aralığıdır.

**low\_port** değişkeni, **port\_range**'deki başlangıç port numarasını tutar. **port\_range** stringi, tire karakterine göre ayrılır ve başlangıç ve bitiş port numaraları iki ayrı string olarak alınır. **int** fonksiyonu kullanılarak, bu stringler tam sayılara dönüştürülür.

**high\_port** değişkeni, **port\_range**'deki bitiş port numarasını tutar. Burada da aynı şekilde **int** fonksiyonu kullanarak stringi tam sayıya dönüştürürüz.

Bu betikteki amaç, belirtilen IP adresindeki belirtilen port aralığındaki açık portları bulmaktır. Bu amaçla, bu değişkenler daha sonra kullanılacaktır.

önceki kısımda açıkladığımız değişkenlere ek olarak, **open\_ports** adında bir liste, **closed\_ports** adında bir sayı ve **lock** adında bir kilitleme değişkeni içerir.

**open\_ports** listesi, tarama sonucunda bulunan açık port numaralarını depolamak için kullanılır.

**closed\_ports** sayısı, kapalı portların sayısını takip etmek için kullanılır.

**lock** değişkeni, birden fazla thread'in aynı anda değişkenlere erişmemesi için kullanılır.

**scan\_port** adında bir fonksiyon tanımlanmıştır. Bu fonksiyon, belirtilen port numarasını tarar ve sonucuna göre **open\_ports** veya **closed\_ports** listelerine veya sayılarına ekler.

**global** anahtar kelimesi, **closed\_ports** değişkeninin global olarak tanımlanmış olduğunu belirtir. Bu sayede fonksiyon içindeki değişiklikler global değişkeni etkiler.

**socket** kütüphanesi kullanılarak bir soket oluşturulur. **s.settimeout(0.5)** satırı, soket işleminin en fazla 0.5 saniye sürmesini sağlar.

**result** değişkeni, **s.connect\_ex((host, port))** satırında belirtilen IP adresi ve port numarası ile bağlantı kurulmasını denemek için kullanılır. **connect\_ex** fonksiyonu, belirtilen port açıksa 0 değerini döndürür. Aksi takdirde, farklı hata kodları döndürür.

Eğer port açıksa, **with lock** bloğu kullanılarak thread'ler arasında çakışmaları önlemek için **open\_ports** listesine port numarası eklenir ve sonuç ekrana yazdırılır. Açık portlar listesi, sonuçları takip etmek için kullanılır.

Port kapalıysa, **closed\_ports** değişkeni artırılır ve sonucu ekrana yazdırmaz.

**threads** listesi, her port için ayrı bir thread oluşturmak için kullanılır.

belirtilen port aralığındaki tüm portları tarar ve sonucuna göre açık veya kapalı olduklarını belirler. Bu tarama işlemi, her bir port için ayrı bir thread oluşturularak yapılır. Tüm thread'lerin tamamlanması beklenir ve ardından sonuçlar ekrana yazdırılır.

**for** döngüsü, belirtilen port aralığındaki her bir port için ayrı bir thread oluşturur. **t = threading.Thread(target=scan\_port, args=(port,))** satırı, **scan\_port** fonksiyonunu ve port numarasını thread'e atanır ve thread başlatılır (**t.start()**). Thread'ler, **threads** listesine eklenir.

**t.join()** satırı, tüm thread'lerin tamamlanmasını bekler. Bu satır, tüm thread'lerin tamamlandığından emin olmak için kullanılır.

**if** koşulu, **open\_ports** listesinin boş olup olmadığını kontrol eder. Eğer açık portlar varsa, **nmap** programına açık port numaralarını ve IP adresini gönderir. **ports\_str = ",".join(str(p) for p in open\_ports)** satırı, **open\_ports** listesindeki tüm açık port numaralarını virgülle ayrılmış bir string'e dönüştürür. **os.system(f"nmap -p {ports\_str} -sV {host}")** satırı, **nmap** programını kullanarak belirtilen portları ve IP adresini taramak için bir sistem komutu çalıştırır.

**else** koşulu, eğer **open\_ports** listesi boş ise, "açık yada böyle bir host yok." şeklinde bir mesaj verir.

Son olarak, **print(Fore.RED + f"Taranan Port: {closed\_ports}")** satırı, tarama sonucunda kapalı olan port sayısını ekrana yazdırır.

**Kullandığımız ve çağırılma şekilleri:**

1. **socket**: Python'un yerleşik soket kütüphanesi, ağ iletişimi ve iletişimde bulunan düşük seviye işlemleri gerçekleştirmek için kullanılır. Bu betikte, soketler bağlantı denemeleri yapmak ve portların durumlarını kontrol etmek için kullanılır.
2. **sys**: Python'un yerleşik sys kütüphanesi, sistemle ilgili parametreler ve işlevlerle çalışmak için kullanılır. Bu betikte, komut satırından girilen argümanları almak ve betiği gerektiğinde sonlandırmak için kullanılır.
3. **os**: Python'un yerleşik os kütüphanesi, işletim sistemi ile ilgili işlevler ve işlemler gerçekleştirmek için kullanılır. Bu betikte, açık portlar için Nmap komutunu çalıştırmak amacıyla kullanılır.
4. **threading**: Python'un yerleşik threading kütüphanesi, betikte eşzamanlı işlemler gerçekleştirmek için kullanılır. Bu betikte, her portu paralel olarak tarayarak tarama sürecini hızlandırmak için kullanılır.
5. **colorama**: Colorama, terminal çıktılarını renklendirmek için kullanılan harici bir kütüphanedir. Bu betikte, terminalde gösterilen banner ve port durumlarını renklendirmek için kullanılır.

**Kütüphanelerin betiğe eklendiği ve çağrıldığı yerler:**

1. İlk önce, betiğin başında **import** anahtar kelimesiyle kütüphaneler eklenir:

import socket

import sys

import os

import threading

from colorama import Fore, Style

1. Kütüphanelerin çağrılması, betiğin ilerleyen kısımlarında gerçekleşir. Örneğin, **socket** kütüphanesi **scan\_port** fonksiyonunda kullanılır:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

1. Örneğin, **sys** kütüphanesi komut satırından argümanları okumak için kullanılır:

host = sys.argv[1]

port\_range = sys.argv[2]

1. **colorama** kütüphanesi, terminalde gösterilen banner ve port durumlarını renklendirmek için kullanılır:

colors = [Fore.RED, Fore.YELLOW, Fore.GREEN, Fore.CYAN, Fore.BLUE, Fore.MAGENTA]

print(colored\_banner + Style.RESET\_ALL)

print(Fore.GREEN + f"Port {port}: Açık" + Style.RESET\_ALL)

+-------------------+

| Başlangıç |

+---------+---------+

|

v

+---------+---------+

| Argümanları kontrol et |

+---------+---------+

|

v

+---------+---------+

| Port aralığını ayarla |

+---------+---------+

|

v

+---------+---------+

| Renkli bannerı yazdır |

+---------+---------+

|

v

+---------+---------+ +----------------+

| Portları tara +-------> Port kapalı|

+---------+---------+ +----------------+

|

v

+---------+---------+ +----------------+

| İş parçacıklarını bekle +----> Port açık |

+---------+---------+ +----------------+

|

v

+---------+---------+

| Sonuçları yazdır |

+---------+---------+

|

v

+---------+---------+

| Nmap ile daha fazla bilgi al |

+---------+---------+

|

v

+---------+---------+

| Son |

+-------------------+