ZAKŁAD SYSTEMÓW ZŁOŻONYCH
Wydział Elektrotechniki i Informatyki
ul. Wincentego Pola 2, 35-959 Rzeszów, tel. 17 865 1340
zsz.prz.edu.pl

WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Dokumentacja projektu

z przedmiotu

ZSiSK

Temat projektu : Network Scanner

Imię i nazwisko	Piotr Dadel
Nr albumu	155687
Rok studiów	III EFZI

1. Wstęp

Projekt polega na utworzeniu skanera sieci który miał za zadanie skanowanie IP oraz portów a na końcu wyświetlenie raportu ze skanowania.

Początkowo do utworzenia skanera wykorzystywałem bibliotekę socket która umożliwiła utworzenie prostego skanera sieci który wyświetlał adresy IP oraz porty ale największym problemem tergo rozwiązania było to że nie wyświetlał wszystkich urządzeń wpiętych do sieci tylko urządzenia „do routera" skanowanie nie było przepuszczane dalej i nie pokazywało wszystkich urządzeń w sieci.

Następnie wykorzystałem bibliotekę Scapy do skanowania adresów IP oraz uzyskiwania adresów MAC oraz bibliotekę socket do uzyskiwania informacji które porty w sieci są otwarte.

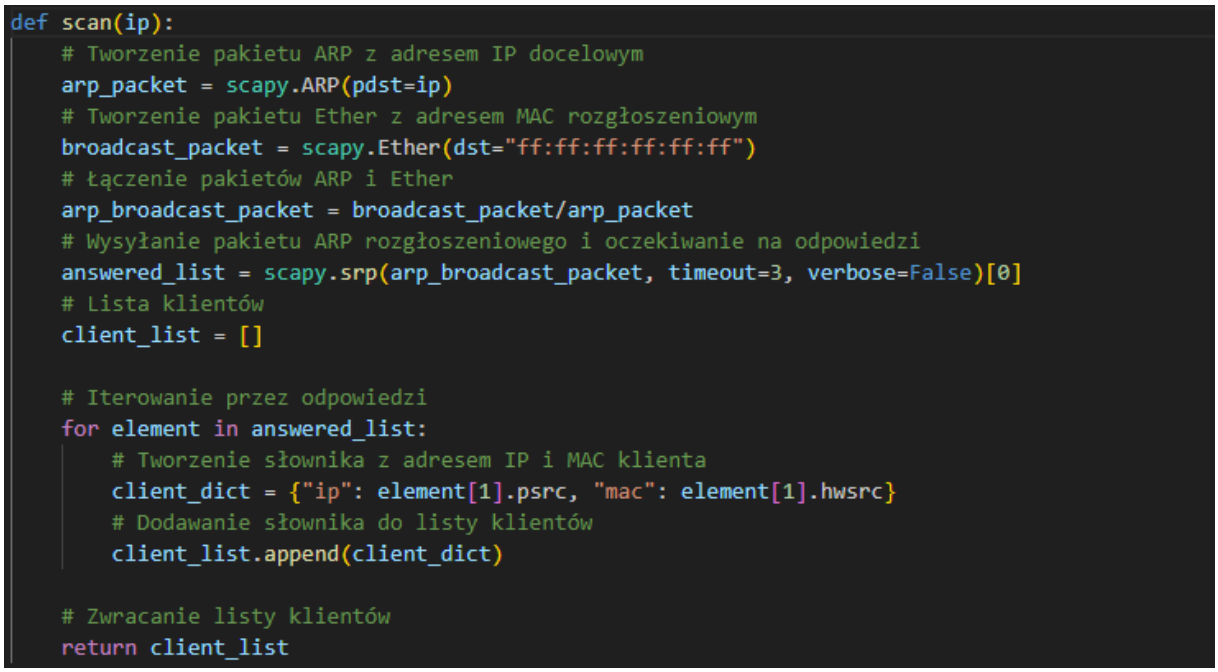
2. Wykonanie

Przy tworzeniu skanera wykorzystano dostępny w Internecie przykład prostego skanera który znalazłem na wielu stronach GitHub oraz artykułach więc ciężko określić kto jest oryginalnym autorem tego rozwiązania które zmodyfikowano do własnych potrzeb i uzupełniono o dodatkowe funkcje jak skaner portów.

Skaner ten umożliwia identyfikację klientów znajdujących się w sieci poprzez wysyłanie pakietów ARP (Address Resolution Protocol) i oczekiwanie na odpowiedzi. Skrypt umożliwia skanowanie pojedynczego adresu IP lub zakresu adresów IP.

W projekcie używamy modułu argparse do obsługi argumentów wiersza poleceń, dzięki czemu użytkownik może podać adres IP lub zakres IP jako argument podczas uruchamiania skryptu. Jeśli użytkownik nie poda adresu IP, skrypt poprosi go o podanie go interaktywnie.

Funkcja get_arguments() służy do parsowania argumentów wiersza poleceń. Tworzymy parser argumentów, dodajemy opcję -t/--target do określania adresu IP lub zakresu IP docelowego. Funkcja ta była zawarta w programie natomiast ja uzupełniłem program o to że w momencie kiedy użytkownik nie podał adresu IP jako argument, skrypt poprosi go o podanie go interaktywnie. Funkcja zwraca przypisane wartości.



Funkcja scan(ip) jest odpowiedzialna za skanowanie sieci. Tworzymy pakiet ARP z adresem IP docelowym, pakiet Ether z adresem MAC rozgłoszeniowym,

a następnie łączymy je w pakiet ARP rozgłoszeniowy. Wysyłamy ten pakiet i oczekujemy na odpowiedzi. Odpowiedzi są zapisywane w liście klientów, gdzie każdy klient jest reprezentowany jako słownik zawierający adres IP i MAC. Funkcja zwraca listę klientów.

W oryginalnym programie wartość oczekiwania na odpowiedz **timeout** była ustawiona na 1 co nie pozwalało na uzyskanie odpowiedzi od niektórych klientów, wydłużyłem ten czas do wartości 3 ponieważ powyżej już nie zauważyłem różnicy a każda modyfikacja tego czasu wydłuża działanie programu.

```
def print_result(scan_list):
    # Drukowanie nagłówka tabeli
    print("IP\t\t\tMAC\n-----------------------------------------")
    # Iterowanie przez listę klientów i drukowanie adresów IP i MAC
    for client in scan_list:
        print(client["ip"] + "\t\t" + client["mac"])

# Pobieranie argumentów
options = get_arguments()
# Wykonanie skanowania
result_list = scan(options.target)
# Drukowanie wyników skanowania
print_result(result_list)
```

Funkcja print_result(scan_list) drukuje wyniki skanowania w postaci tabeli, wyświetlając adresy IP i MAC klientów. Iterujemy przez listę klientów i drukujemy informacje dla każdego z nich.

Na końcu skryptu pobieramy argumenty z użyciem funkcji get_arguments(), wykonujemy skanowanie wywołując funkcję scan(options.target) i drukujemy wyniki skanowania używając funkcji print_result(result_list)

```python
print ("Skanowanie portów")
#skanowanie portów za pomocą socket
def scan_port(ip, port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)
        result = sock.connect_ex((ip, port))
        if result == 0:
            return True
        else:
            return False
    except Exception:
        return False

def scan_network(network):
    ip_prefix = network.split('.')[0:3]
    for i in range(1, 255):
        ip = '.'.join(ip_prefix + [str(i)])
        for port in range(1, 8080):  # Skanujemy porty od 1 do 8080
            if scan_port(ip, port):
                print(f"IP: {ip}, Port: {port} - OPEN")

network = options.target  # Pobranie adresów do skanowania z cześniej wpisanego adresu
scan_network(network)
```

Ostatnim elementem programu jest skaner portów utworzony za pomocą biblioteki socket. Problemem tej części programu jest bardzo wolne działanie ponieważ skanuje on każdy możliwy port z osobna co zajmuje dużo czasu, podczas działania programu można zaobserwować dwa początkowe porty (53 i 80) co już wystarczy aby stwierdzić czy program działa a jeżeli zadanie wymaga dokładnego prześledzenia otwartych portów poczekać na zakończenie działania programu. W programie możemy zmieniać zakres portów jaki ma przeskanować co może skrócić jego działanie i dostosować go do naszych potrzeb.

3. Przykład działania programu

Pierwsza część programu po wpisaniu jaką podsieć i w jakim zakresie (/24) ma przeskanować szybko przystępuje do działania i wyświetla uzyskany raport w zrzucie ekranu poniżej przykład skanowania mojej sieci w momencie testów programu.

```
Podaj adres IP sieci np.(192.168.0.0/24): 192.168.50.0/24
IP                    MAC
----------------------------------------
192.168.50.1          50:eb:f6:85:51:b8
192.168.50.50         1c:c1:0c:df:08:f9
192.168.50.26         a4:55:90:6c:b5:88
192.168.50.95         04:03:d6:53:9f:87
192.168.50.208        58:fd:b1:db:18:b9
PS C:\Users\piotr\Desktop\ZSISK-Projekt>
```

Wraz z drugą częścią programu efekt skanowania wygląda następująco, po wyświetleniu dwóch dostępnych portów program został zatrzymany.

```
Podaj adres IP sieci np.(192.168.0.0/24): 192.168.50.0/24
IP                    MAC
----------------------------------------
192.168.50.1          50:eb:f6:85:51:b8
192.168.50.50         1c:c1:0c:df:08:f9
192.168.50.95         04:03:d6:53:9f:87
192.168.50.208        58:fd:b1:db:18:b9
Skanowanie portów
IP: 192.168.50.1, Port: 53 - OPEN
IP: 192.168.50.1, Port: 80 - OPEN
PS C:\Users\piotr\Desktop\ZSISK-Projekt>
```