Wydział Informatyki i

Telekomunikacji

Informatyka

Podstawy Teleinformatyki - projekt

WireDolphin - Graficzny interfejs manipulatora pakietów

Studia stacjonarne I stopnia

Semestr 6.

Artur Bałczyński

Mateusz Ostrowski

Adam Przywuski

Grupa L4, Zespół 4

11 września 2020

Rozwiązanie dostępne na repozytorium:

https://github.com/matostr98/packet-manipulator

Opis aplikacji

WireDolphin

Aplikacja *WireDolphin* powstała jako projekt w ramach przedmiotu Podstawy Teleinformatyki. Tematem projektu brzmiał: graficzny interfejs manipulator pakietów. Aplikacja działa dla wszystkich interfejsów znajdujących się w systemie. Dla każdego z interfejsów wyświetla przesyłanie dla niego pakiety warstw od dostępu do sieci do aplikacji. Każdy pakiet można edytować i retransmitować. Aplikacja działa szybko i wydajnie dzięki zastosowaniu wielowątkowości, a wszystkie operacje są niezwykle proste do wykonania dzięki wygodnemu

interfejsowi. Przeznaczona jest ona głównie dla administratorów sieci. Inspiracją do tej aplikacji był program *WireShark*. *WireDolphin* pozwala jednak na bardzo proste dodawanie nowych modułów i funkcjonalności dzięki zastosowaniu łatwego w nauce i programowaniu języka Python.

Podział prac

Podział prac przy aplikacji wyglądał następująco:

Imię i nazwisko	**Zadania**
Artur Bałczyński	● Wyszukiwanie interfejsów ● Wielowątkowość ● Sniffer ● Dokumentacja
Mateusz Ostrowski	● Wyświetlanie interfejsów ● Interfejs dla pakietów ● Wielowątkowość ● Dokumentacja
Adam Przywuski	● Edycja pakietów ● Model pakietu ● Interfejs edycji pakietów ● Dokumentacja

Funkcjonalność aplikacji

- Wykrywanie dostępnych interfejsów
    - Wyświetla wszystkie dostępne interfejsy (również te z HyperV)
    - Pokazuje wszystkie istotne informacje o wykrytych interfejsach takie jak adresy IPv4, IPv6, MAC
- Podsłuchiwanie pakietów
    - Podsłuchuje wszystkie pakiety w ramach wybranego interfejsy
    - Wyświetla informacje o źródłowym i docelowym IP
    - Wyświetla zdefiniowane protokoły np. TCP, UDP, HTTP, DNS
    - Pozwala wyświetlić dane przenoszone przez pakiet
    - Pozwala na wstrzymanie i kontynuację podsłuchiwania
    - Pozwala na zmianę interfejsu
    - Podsłuchiwanie nie blokuje interfejsu
- Edycja pakietów
    - Pozwala na ponowne wysłanie pakietu zarówno oryginalnego jak i edytowanego
    - Pozwala na edycję adresów źródłowych i docelowych IP, protokołu, portu oraz danych
- Interfejs graficzny
    - Ułatwia posługiwanie się aplikacją
    - Posiada wszystkie niezbędne elementy takie jak przyciski aktywujące poszczególne funkcje oraz paski przewijania
- Asynchroniczność
    - Pozwala na szybkie i wydajne działanie progamu
    - Jeden wątek podsłuchuje pakiety i umieszcza je w kolejce
    - Drugi wątek odbiera pakiety i wyświetla je w interfejsie

Wybrane technologie

Python

Python jest interpretowanym językiem programowania wysokiego poziomu i ogólnego przeznaczenia. Stworzona przez Guido van Rossuma i wydana po raz pierwszy w 1991 roku, filozofia projektowania Pythona kładzie nacisk na czytelność kodu dzięki znacznemu wykorzystaniu znacznych białych znaków(takich jak tzw. wcięcia w kodzie). Jego konstrukcje językowe i podejście obiektowe mają na celu pomóc programistom w pisaniu jasnego, logicznego kodu dla małych i dużych projektów.

Obsługuje wiele paradygmatów programowania, w tym programowanie strukturalne (szczególnie proceduralne), obiektowe i funkcjonalne. Python jest często opisywany jako język „z bateriami” ze względu na obszerną bibliotekę standardową.

Python powstał pod koniec lat 80-tych jako następca języka ABC. Python 2.0, wydany w 2000 roku, wprowadził funkcje, takie jak listy składane i system czyszczenia pamięci z liczeniem referencji.

Python 3.0, wydany w 2008 roku, był główną wersją języka, który nie jest w pełni kompatybilny wstecz, a większość kodu Pythona 2 nie działa bez modyfikacji w Pythonie 3.

Język Python 2 został oficjalnie wycofany w 2020 r. (Po raz pierwszy planowano na 2015 r.), A „Python 2.7.18 jest ostatnią wersją Pythona 2.7, a zatem ostatnią wersją Pythona 2". Nie będą już wydawane żadne poprawki zabezpieczeń ani inne ulepszenia za to. Wraz z wycofaniem Pythona 2 obsługiwane są tylko Python 3.5.x i nowsze.

Interpretery Pythona są dostępne dla wielu systemów operacyjnych. Globalna społeczność programistów opracowuje i utrzymuje CPython, darmową implementację referencyjną typu open source . Organizacja non-profit, Python Software Foundation, zarządza zasobami związanymi z programowaniem w językach Python i CPython i kieruje nimi.

W projekcie został użyty Python 3.7 ze względu na fakt, że jest on najnowszą dostepną wersją oraz bezproblemowości w działaniu z użytymi bibliotekami. W systemie użyto popularnych struktur danych takich jak:

-Klasa to szablon kodu służący do tworzenia obiektów. Obiekty mają zmienne składowe i są z nimi powiązane zachowanie. W Pythonie klasa jest tworzona przez słowo kluczowe class. Obiekt tworzony jest za pomocą konstruktora klasy. Ten obiekt będzie wtedy nazywany instancją klasy. Został użyty w celu przechowywania parametrów odebranych pakietów jak i interfejsów dostępnych na urządzeniu użytkownika.

-Lista jest kontenerem, który może pomieścić stałą liczbę elementów, które powinny być tego samego typu. Większość struktur danych wykorzystuje tablice do implementacji swoich algorytmów. Składa się z elementu oraz indeksu, te 2 wartości są ściśle między sobą powiązane i dzięki nim mamy możliwość. W systemie zostały użyte w celu przechowywania zdobytych już pakietów.

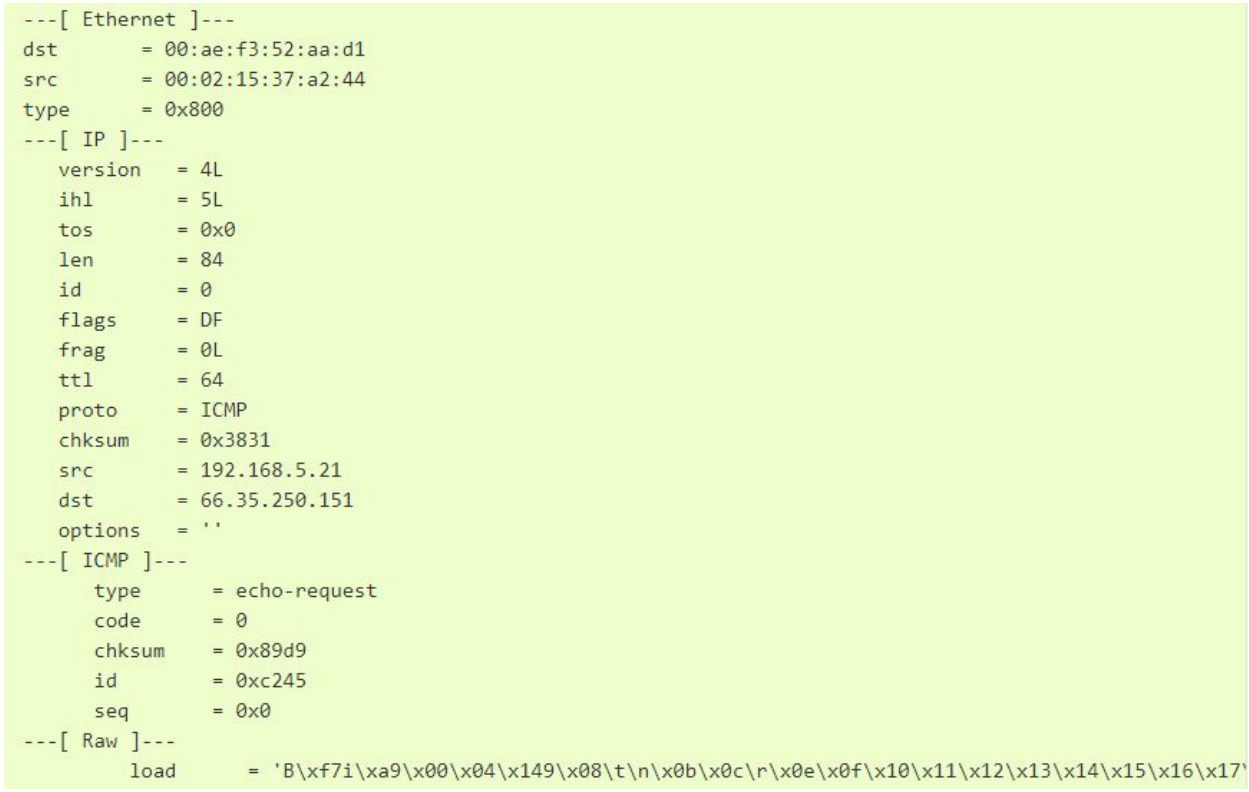
Python Virtual Environment

Środowisko wirtualne to narzędzie, które pomaga utrzymać zależności wymagane przez różne projekty, tworząc dla nich izolowane środowiska wirtualne Pythona. Jest to jedno z najważniejszych narzędzi, z którego korzysta większość programistów Pythona.
Domyślnie każdy projekt w systemie będzie używał tych samych katalogów do przechowywania i pobierania pakietów witryn (bibliotek innych firm). Jakie to ma znaczenie? Teraz, w powyższym przykładzie dwóch projektów, masz dwie wersje biblioteki tkinter. Jest to prawdziwy problem dla Pythona, ponieważ nie może on rozróżniać wersji w katalogu „site-packages”. Tak więc zarówno wersja 1.9, jak i wersja 1.10 będą znajdować się w tym samym katalogu o tej samej nazwie. W tym miejscu do gry wchodzą środowiska wirtualne. Aby rozwiązać ten problem, musimy po prostu utworzyć dwa oddzielne środowiska wirtualne dla obu projektów. Wspaniałą rzeczą jest to, że nie ma ograniczeń co do liczby środowisk, które można mieć, ponieważ są to tylko katalogi zawierające kilka skryptów.

Scapy

Scapy to potężny, oparty na Pythonie, interaktywny program i biblioteka do obsługi pakietów. Jest w stanie sfałszować lub dekodować pakiety wielu protokołów, przesyłać je przewodowo, przechwytywać, przechowywać lub odczytywać przy użyciu plików pcap, dopasowywać żądania i odpowiedzi i wiele więcej. Został zaprojektowany, aby umożliwić szybkie prototypowanie pakietów przy użyciu domyślnych, które działają.
Z łatwością radzi sobie z większością klasycznych zadań, takich jak skanowanie, śledzenie tras, sondowanie, testy jednostkowe, ataki lub wykrywanie sieci (może zastąpić hping, 85% nmap, arpspoof, arp-sk, arping, tcpdump, wireshark, p0f itp.). Działa również bardzo dobrze w wielu innych konkretnych zadaniach, których większość innych narzędzi nie może obsłużyć, takich jak wysyłanie nieprawidłowych ramek, wstrzykiwanie własnych ramek 802.11, łączenie technik (przeskakiwanie VLAN + zatruwanie pamięci podręcznej ARP, dekodowanie VoIP na kanale chronionym WEP,. ..) itp. Wszystkie pakiety opierają się o tzw. Warstwy gdzie każdy poziom odpowiada za inna wartstwę, jest to bardzo łatwy sposób umożliwiający na dokładną kontrolę

pakietów.



```
---[ Ethernet ]---
dst       = 00:ae:f3:52:aa:d1
src       = 00:02:15:37:a2:44
type      = 0x800
---[ IP ]---
   version   = 4L
   ihl       = 5L
   tos       = 0x0
   len       = 84
   id        = 0
   flags     = DF
   frag      = 0L
   ttl       = 64
   proto     = ICMP
   chksum    = 0x3831
   src       = 192.168.5.21
   dst       = 66.35.250.151
   options   = ''
---[ ICMP ]---
      type      = echo-request
      code      = 0
      chksum    = 0x89d9
      id        = 0xc245
      seq       = 0x0
---[ Raw ]---
         load      = 'B\xf7i\xa9\x00\x04\x149\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17'
```

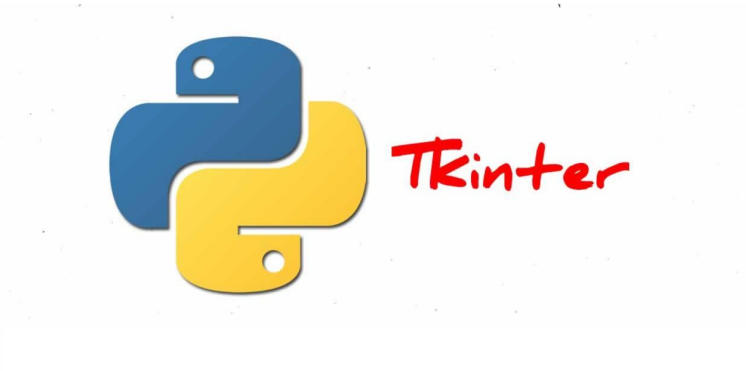
Zdjęcie przedstawiające podział pakietu na warstwy. Wyróżniamy tu 3 warstwy: Ethernet, IP, ICMP. Każda warstwa dzieli pakiet na części, w której występują określone pola predefiniowane dla danej warstwy. Warstwa nr 3 jest to warstwa protokołowa, która jest tą najczęściej zmienną i tam należy spodziewać się największych zmian między pakietami.

Pomimo braku jakiekolwiek wartościowej dokumentacji (większość dokumentacji opiera się o jeden przykład nie rozkładając tego na inne protokoły) w miarę intuicyjna umożliwiająca bogate możliwości w dziedzinie pakietów internetowych. Jest rozbudowana, a jednocześnie nie posiada żadnych rażących błędów, które by uniemożliwiały dalszą pracę.
W systemie zostały użyte następujące funkcje, struktury z danej biblioteki:

- sniff(filter=””iface=””,count=1)- jest to funkcja służąca za podsłuchiwanie pakietów uzyskanych bezpośrednio na danym interfejsie. Zwraca ona właśnie uzyskane te pakiety. Argumenty
    - filter= służy do filtrowania pakietów. Zostaną przechwycone tylko te, które zawierają słowo kluczowe z filtra.

    - iface= służy do wyboru interfejsu, z którego pakiety mają zostać przechwycone. Jeśli nie występuje te pole domyślnie to są wszystkie.
    - count=jest to liczba ile ma być odebranych pakietów w ciągu działania jednego callbacka funkcji. Pierwotnie ta liczba wynosi 1.
- sendp(Layer1/Layer2/Layer3)- funkcja służąca do wysyłania pakietów w warstwie 2, jest to funkcja wystarczająco do przesłania pakietu.
    - Layer1 =jest to warstwa Ethernetu
    - Layer2 = jest to warstwa IP
    - Layer3 =jest to warstwa protokołów takich jak TCP/UDP itd.
- get_windows_if_list()-funkcja służąca za pobranie wszystkich interfejsów występujących w systemie.

Tkinter



Tkinter to biblioteka Pythona powiązana z zestawem narzędzi Tk GUI. Jest to standardowy interfejs Pythona do zestawu narzędzi Tk GUI, i jest de facto standardowym interfejsem GUI Pythona. Tkinter jest dołączony do standardowych instalacji Pythona w systemach Linux, Microsoft Windows i Mac OS X.

Nazwa Tkinter pochodzi od interfejsu Tk. Tkinter został napisany przez Fredrika Lundha. Jest to najpopularniejsza biblioteka do tworzenia interfejsu w języku Python, wynika to z łatwej obsługi i dobrze stworzonej dokumentacji. Warto jednak zauważyć, że jest ona dosyć przestarzała i niestety nie jest możliwe utworzenie w niej rozbudowanego i nowoczesnego interfejsu do aplikacji. Jego zadanie jest głównie skierowane do prostych interfejsów nie rozbudowanych projektów, gdzie nie zalecane jest korzystanie z zewnętrznych języków do tworzenia frontendu. W systemie zostały wykorzystane następujące struktury-

- Tk()-funkcja służąca za utworzenia okna, gdzie są dodawane pozostałe widżety.
- Label()-Jest to pole gdzie jest wyświetlany dany tekst.
- Entry()-Jest to pole gdzie można wpisywać własny tekst
- Button()- Jest to przycisk odpowiedzialny za aktywowanie funkcji.
- Scrollbar- Umożliwia łatwą nawigację po oknie.
- Treeview- jest to rodzaju lista, która umożliwia wpisywanie parametrów struktury.

Git

Git jest rozproszonym systemem kontroli wersji będący oprogramowaniem typu open source. Jest on opublikowany na licencji GNU GPL 2. Stworzony był przez Linusa Torvaldsa w celu wspomożenia rozwoju jądra Linux. Ułatwia pracę nad projektami tworzonymi przez wielu programistów, pozwalając na jednoczesną pracę nad wieloma funkcjonalnościami oraz ich łatwe

łączenie. Dodatkowo zapamiętuje rewizje co pozwala na przywrócenie działającej funkcjonalności.

Github

Github jest hostingowym serwisem internetowym, przeznaczonym przede wszystkim dla programistów. Wykorzystuje system kontroli wersji Git. Pozwala na darmowe hostowanie programów. Zapewnia wszystkie standardowe funkcje systemu Git, rozszerzając je o własne takie jak protekcja gałęzi, organizacje, kontrolę dostępu, hosting stron internetowych.

JetBrains PyCharm

Jeden z najbardziej popularnych środowisk programistycznych dla języka Python. Jest częścią ekosystemu JetBrains oferującego środowiska programistyczne dla wielu bardziej i mniej popularnych języków programowania. Tak jak inne produkty JetBrains posiada podobny interfejs który pozwala nie tylko na edycję kodu, ale również łatwe i precyzyjne wykrywanie błędów i ich naprawianie. Posiada funkcjonalności systemów kontroli wersji pozwalając na bezproblemową i graficzną komunikację z serwisami hostingującymi.

Architektura programu

Program stanowi jednolitą aplikację okienkową, samodzielnie działającą w środowisku Python, na systemach operacyjnych Windows, Linux oraz MacOS. Aplikacja podzielona została na:

- Interfejs graficzny użytkownika - umożliwiający łatwe odczytywanie oraz edycję danych pakietów internetowych, a także przeglądanie oraz wybór dostępnych interfejsów sieciowych. Zawiera elementy takie jak:
    - Okna systemowe
    - Etykiety
    - Przyciski
    - Listy hierarchiczne
    - Edytowalne pola tekstowe
- Moduły funkcjonalne odpowiedzialny za:
    - Przechwytywanie informacji o pakietach, takich jak:
        - Adres źródłowy
        - Adres docelowy
        - Protokół
        - Długość
        - TTL
    - Zbieranie informacji o dostępnych interfejsach sieciowych, w tym:
        - Nazwy

        - Adresu IPv4
        - Adresu IPv6
        - Opisu systemowego
    - Modyfikację danych pakietów, w tym:
        - Adresu źródłowego
        - Adresu docelowego
        - Protokołu
        - Zawartości pakietu
- Moduł obsługujący wątki:
    - Producenta - przechwytującego pakiety i dodającego je do kolejki
    - Konsumenta - odczytującego pakiety z kolejki i wyświetlającego je na cyklicznie odświeżonej liście w interfejsie graficznym

  oraz synchronizację między nimi
- Klasy przechowując modele przetwarzanych interfejsów sieciowych oraz pakietów, takich jak:
    - Warstwa 1:
        - Ethernet
    - Warstwa 2:
        - IP
    - Warstwa 3:
        - TCP
        - UDP
    - Warstwa 4:
        - FTP
        - SSH
        - Telnet
        - SMTP
        - Telnet
        - DNS

        - DHCP
        - HTTP
        - HTTPS
- Globalne parametry filtrujące przechwytywane pakiety, w tym:
    - Interfejs sieciowy
    - Protokoły sieciowe
    - Adres źródłowy
    - Adres docelowy

  Działanie programu można przedstawić za pomocą diagramu stanów:

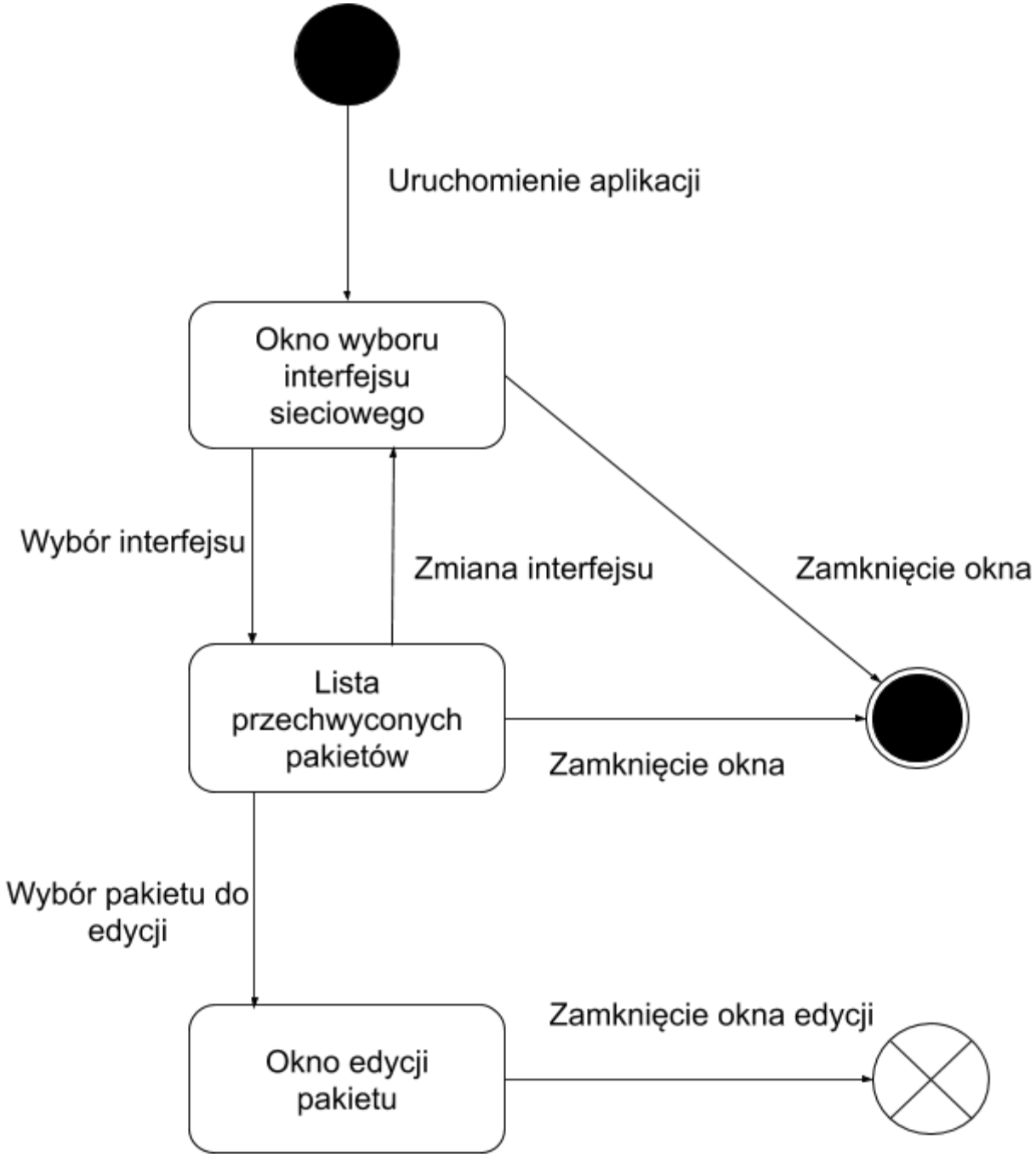


Diagram stanów aplikacji

Interesujące problemy i rozwiązania

Podstawowym napotkanym problemem było jednoczesne przechwytywanie pakietów oraz wyświetlanie na cyklicznie odświeżanej liście, przy założeniu że równolegle powinny działać inne funkcje aplikacji takie jak: wybór oraz edycja pakietu. W celu rozwiązania tego problemu wprowadzono wielowątkowość. Aplikacja działa wykorzystując trzy wątki:

1. Wątek główny - realizujący działanie elementów interfejsu graficznego oraz wybór pakietu do edycji
2. Wątek producenta - przechwytującego pakiety oraz dodającego je do kolejki
3. Wątek konsumenta - pobierającego pakiety z kolejki oraz wyświetlającego informacje o nich na cyklicznie odświeżanej liście

Pracę wątków można zobrazować na diagramie:

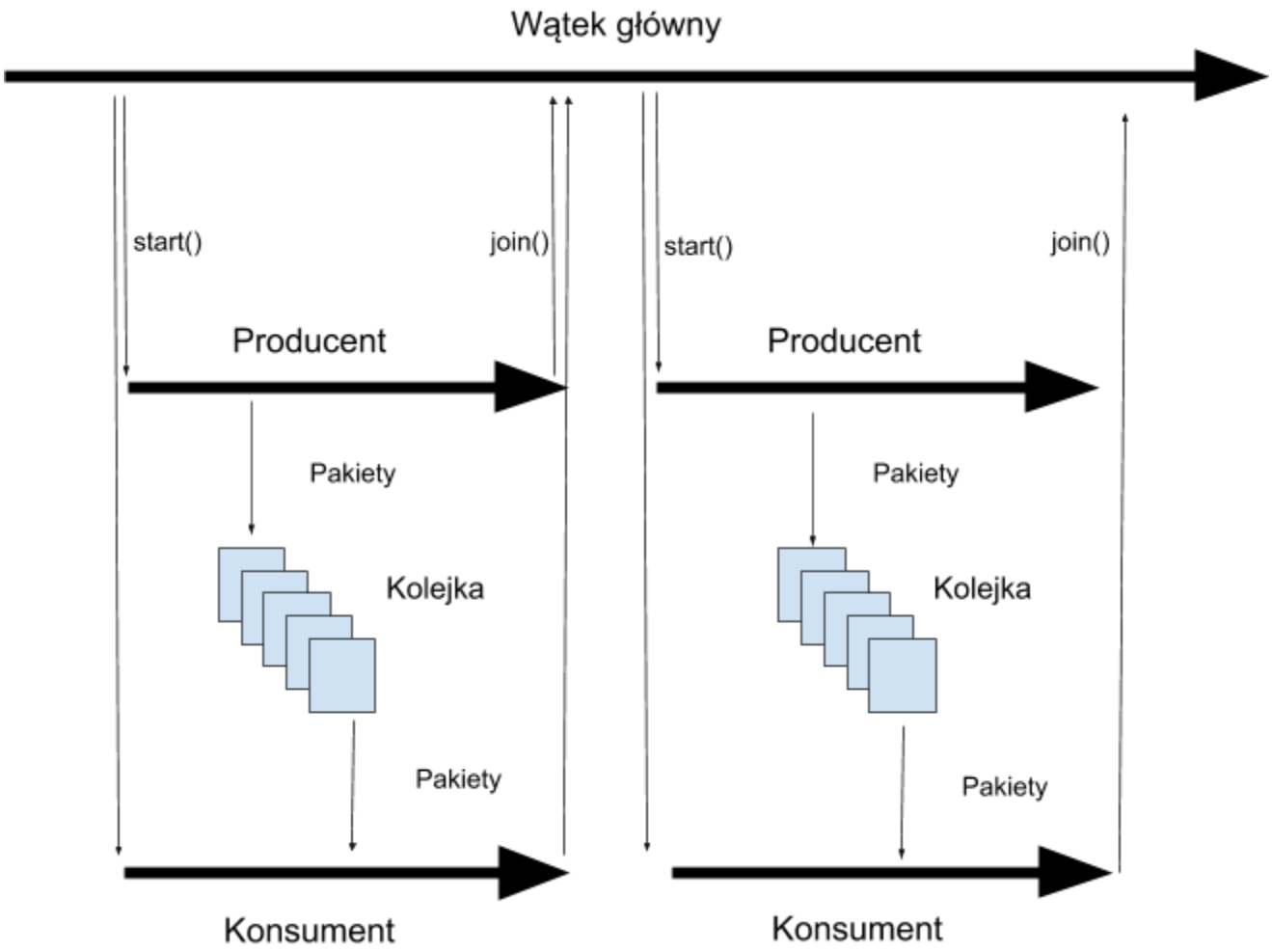


Diagram obrazujący pracę wątków

Wątki producenta i konsumenta tworzone są przy otwieraniu ekranu wyświetlania przechwyconych pakietów, po wyborze interfejsu sieciowego. Wątki mogą zostać wstrzymane i wznowione za pomocą przycisku “Pasuse/Resume”; wstrzymanie spowoduje zaprzestanie przechwytywania i dodawania do wyświetlonej listy kolejnych pakietów. Wątki są zatrzymywane również przy ponownym przejściu do ekranu wyboru interfejsu sieciowego oraz wznawiane po jego wybraniu. Dzięki takiemu rozwiązaniu pakiety są przechwytywane i wyświetlane w czasie rzeczywistym przy zachowaniu możliwości równoległego korzystania z funkcji aplikacji takich jak: wybieranie oraz edycja pakietu.

Działanie wątków producenta i konsumenta realizują następujące fragmenty kodu:

Kod producenta - plik: *producer_thread.py*

```
import logging
import threading

from scapy.sendrecv import sniff

from gui.parameters.filter_parameters import FilterParameters
from gui.parameters.global_parameters import GlobalParameters
from model.ip_packet import IpPacket

logging.basicConfig(level=logging.DEBUG,
                    format='(%(threadName)-9s) %(message)s', )



class ProducerThread(threading.Thread):
    def __init__(self, the_queue):
        super().__init__()
        self.the_queue = the_queue

    def run(self):
        while True:
            self.sniff_packets()
            if GlobalParameters.stop_thread_flag:
                break

    def sniff_packets(self):
        packet = sniff(iface=FilterParameters.interface,
filter='ip', count=1)
```

```
        ip_packet = IpPacket(packet[0])
        logging.debug('create ' + packet[0].summary())
        self.the_queue.put(ip_packet)
```

Kod producenta - plik: consumer_thread.py

```
import logging
import threading
import tkinter as tk

from gui.parameters.global_parameters import GlobalParameters
from model.ip_packet import IpPacket

logging.basicConfig(level=logging.DEBUG,
                    format='(%(threadName)-9s) %(message)s', )


class ConsumerThread(threading.Thread):

    def __init__(self, packets_queue, table, packets_list):
        super().__init__()
        self.the_queue = packets_queue
        self.table = table
        self.packets_list: list = packets_list

    def run(self):
        while True:
            self.refresh_data()
```

```
            if self.the_queue.empty() and
GlobalParameters.stop_thread_flag:
                break


    def refresh_data(self):
        while not self.the_queue.empty():
            data: IpPacket = self.the_queue.get()
            logging.debug('get ' + str(data) + ' ' +
str(GlobalParameters.packet_index))
            self.packets_list.append(data)
            index = len(self.packets_list) - 1
            self.table.insert("", tk.END, index, text=index,
                              values=(data.source,
                                      data.destination,
                                      data.protocol,
                                      data.length,
                                      data.ttl))
            self.table.update_idletasks()
```

Kolejnym bardzo uciążliwym problemem było znalezienie odpowiedniej dokumentacji scapy odpowiedzialnej za poszczególne pakiety. Na stronie głównej biblioteki: https://scapy.readthedocs.io/en/latest/index.html znajduje się niewystarczająca liczba informacji. Jednym z takich powodów jest fakt, że scapy jest to jednocześnie biblioteka jak i oprogramowanie, co powoduje, że większość informacji jest skierowane w strone użytkownika oprogramowania a nie bezpośrednio programisty, co wiąże się z niewystarczającą ilością przykładów i prawie zerowym opisem bezpośrednio funkcji. Dlatego wiele fragmentów kodu było to eksperymentowanie i sprawdzanie różnych możliwości. Warto też podkreślić pomoc

innych portali dedykowanych dla programistów, którzy również spotkali się z analogicznymi problemami.

Instrukcja użytkowania aplikacji

Instalacja oprogramowania:

Żeby zainstalować Python trzeba wybrać określoną wersję z danej strony https://www.python.org/downloads/windows/ i podążać zgodnie z instalacją.

Następnie jeśli oprogramowania zostało zainstalowane warto sprawdzić czy wszystko jest poprawne.Dlatego warto sprawdzić poprzez otworzenie konsoli i wpisanie “Python”.

Jeśli instalacja się powiodła pojawi się informacja o pobranej wersji.

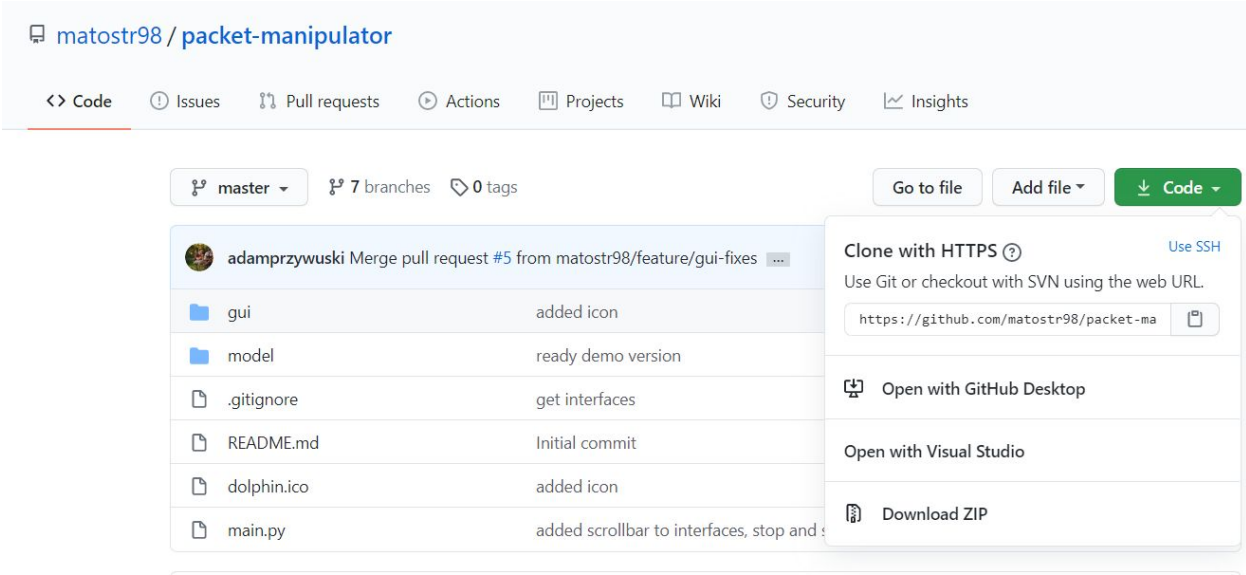
```
C:\Users\adamp>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

W następnym kroku trzeba zainstalować potrzebne biblioteki, z których system korzysta

```
C:\Users\adamp> pip install --pre scapy[basic]
Requirement already satisfied: scapy[basic] in c:\users\adamp\appdata\roaming\python\python37\site-packages (2.4.3)
Collecting ipython
  Downloading https://files.pythonhosted.org/packages/72/36/89e1bb437f4f2275c33acc6eb333ab2d1c64e732ad23d6f34825b512e1a3
/ipython-7.18.1-py3-none-any.whl (786kB)
     |████████████████████████████████| 788kB 139kB/s
Collecting jedi>=0.10
  Downloading https://files.pythonhosted.org/packages/c3/d4/36136b18daae06ad798966735f6c3fb96869c1be9f8245d2a8f556e40c36
/jedi-0.17.2-py2.py3-none-any.whl (1.4MB)
     |████████████████████████████████| 1.4MB 123kB/s
Collecting prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0
  Downloading https://files.pythonhosted.org/packages/2b/c1/53ac685833200eb77ef485c2220dac5bfc255418e660790a9eb5cf3abf25
/prompt_toolkit-3.0.7-py3-none-any.whl (355kB)
     |████████████████████████████████| 358kB 139kB/s
Requirement already satisfied: setuptools>=18.5 in c:\users\adamp\appdata\roaming\python\python37\site-packages (from ip
ython->scapy[basic]) (46.4.0)
Collecting traitlets>=4.2
  Downloading https://files.pythonhosted.org/packages/e6/20/f73a1130598ffaf561fda74d761bbe66db7a3639f34456761e781509881b
/traitlets-5.0.4-py3-none-any.whl (98kB)
     |████████████████████████████████| 102kB 126kB/s
Collecting pygments
  Downloading https://files.pythonhosted.org/packages/2d/68/106af3ae51daf807e9cdcba6a90e518954eb8b70341cee52995540a53ead
/Pygments-2.6.1-py3-none-any.whl (914kB)
     |████████████████████████████████| 921kB 139kB/s
Requirement already satisfied: decorator in d:\python\lib\site-packages (from ipython->scapy[basic]) (4.4.0)
Collecting backcall
```

Instalacja programu i uruchomienie

W celu pobrania programu trzeba wejść na stronę: https://github.com/matostr98/packet-manipulator i kliknąć zielony przycisk “Code”. Tam pojawią się zakładki z których wybieramy Download ZIP

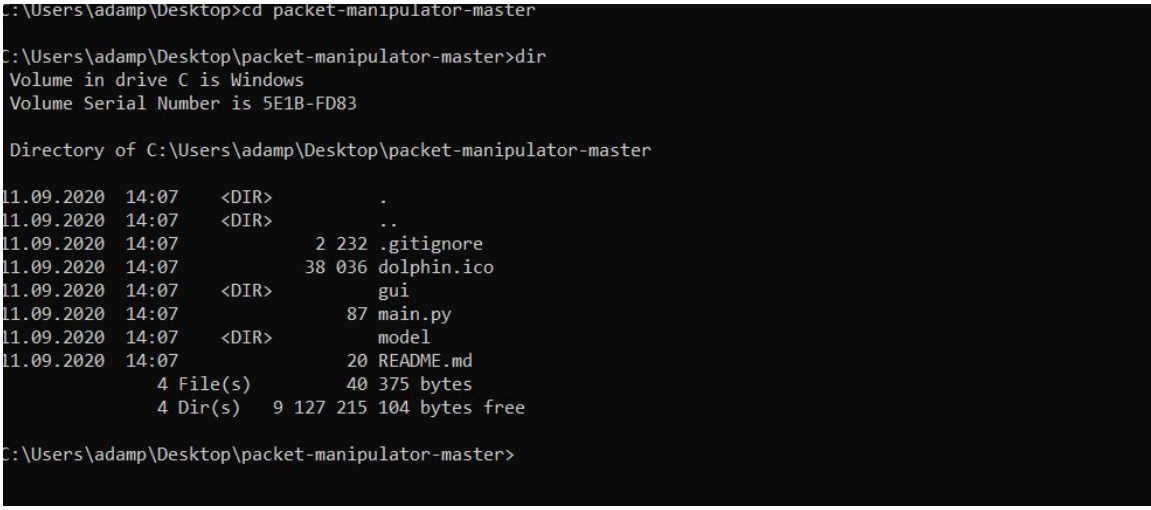


Po pobraniu pliku wypakowujemy go w dowolnie wybranym miejscu w systemie.

Można również sklonować repozytorium do wybranego folderu za pomocą komendy

```
git clone
https://github.com/matostr98/packet-manipulator.git
```

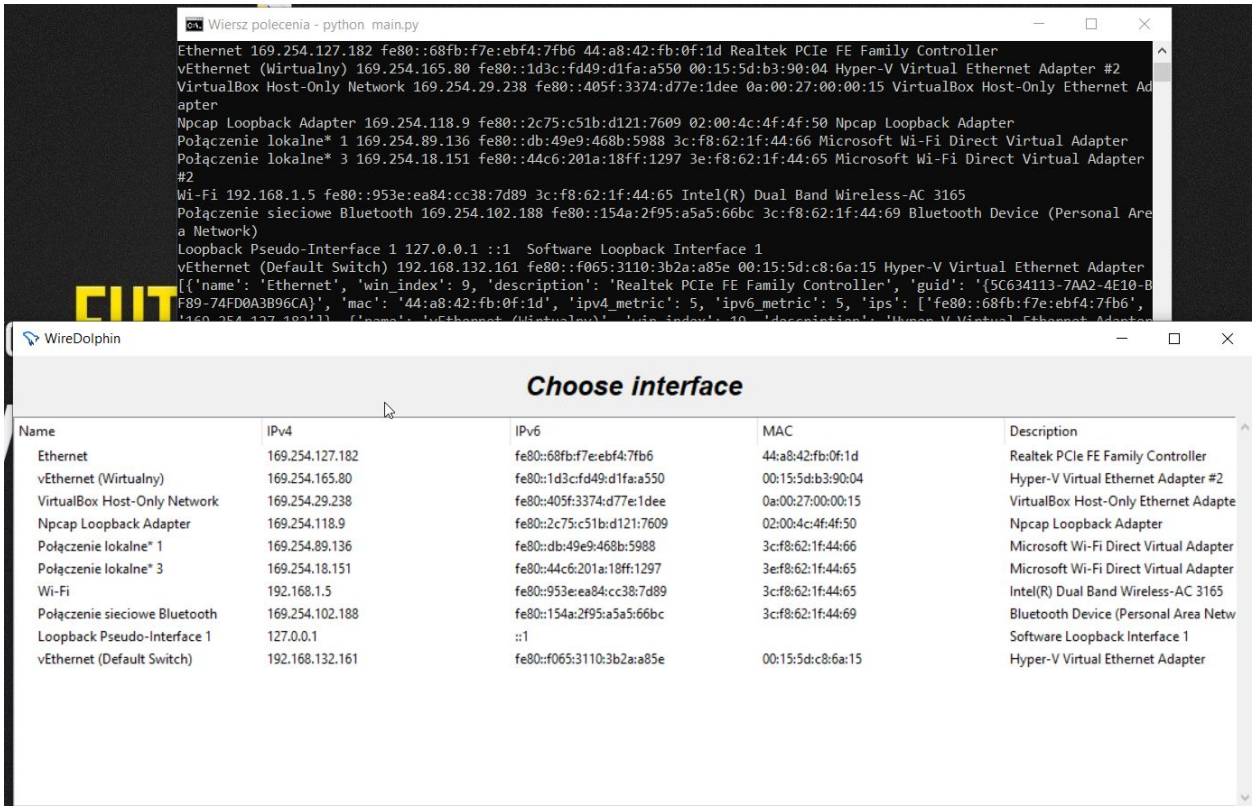
Następnie otwieramy konsole i za pomocą odpowiednich komend przechodzimy do rozpakowanego lub sklonowanego folderu w którym znajduje się repozytorium projektu.



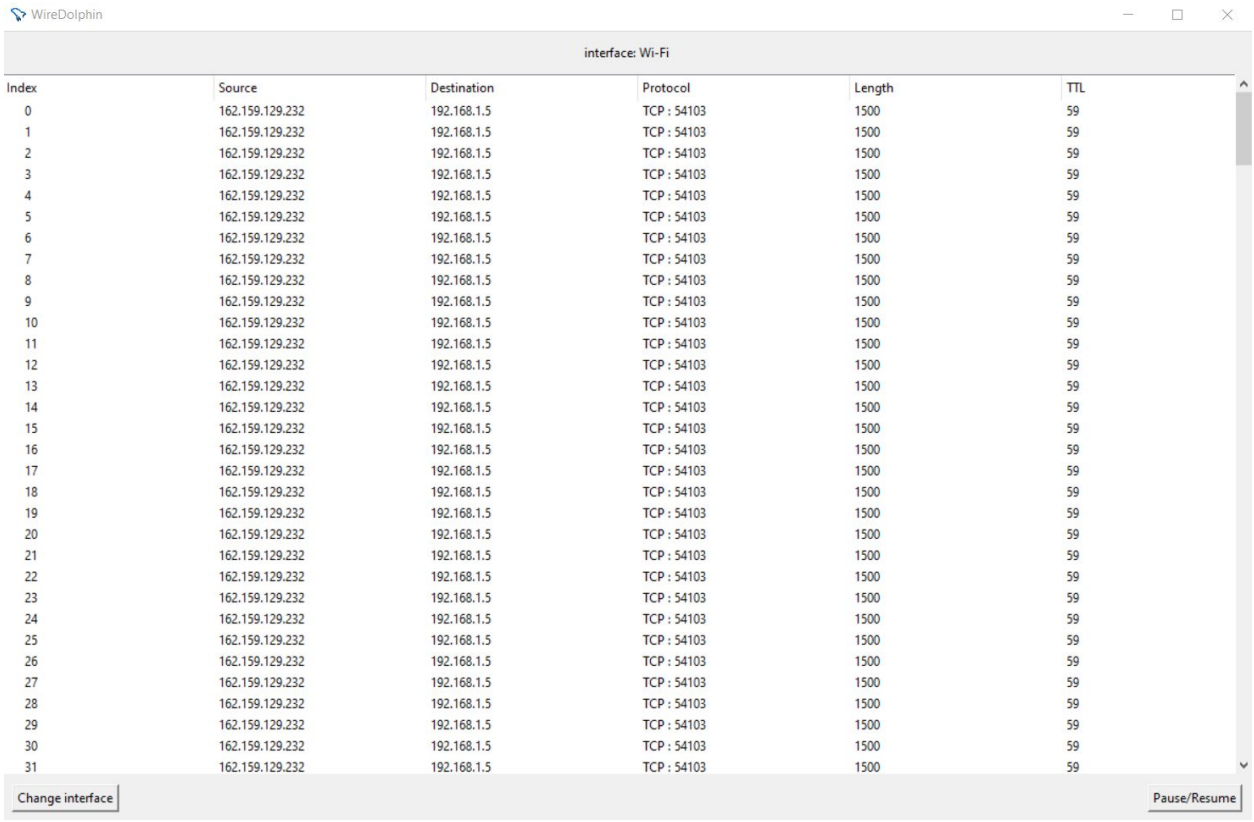
Żeby uruchomić program wpisujemy `python main.py`

Użytkowanie systemu

Po włączeniu programu powinno nam wyskoczyć okno, a w konsoli powinny zostać wypisane informacje dotyczące interfejsów.



Jest to pierwsze okno, które odpowiada za wybranie interfejsu sieciowego, który chcesz podsłuchać. Warto pamiętać, że na screenie występują tylko wybrane interfejsy, u każdego ilość i rodzaje mogą wyglądać kompletnie inaczej więc warto o tym pamiętać. Żeby wybrać odpowiedni interfejs trzeba na niego kliknąć dwukrotnie. Następnie zostanie przeniesieni do okna podsłuchiwania.



Następne okno wygląda bardzo zbliżenie do poprzedniego z różnicą taką, że występują tutaj pakiety zamiast interfejsów. Są one wyświetlane wraz z najważniejszymi informacjami:

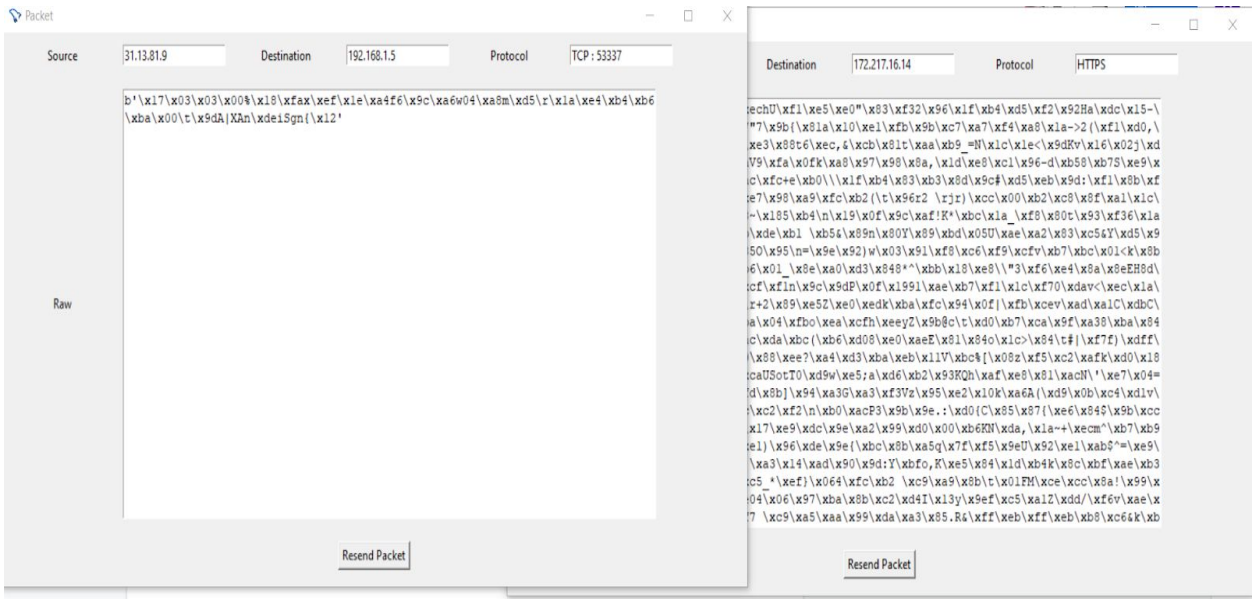
- Index- numer pakietu
- Source- oznacza adres Ip z którego został nadesłany pakiet
- Destination- oznacza adres Ip docelowy do którego ma być dostarczony
- Protocol- jest to pole, gdzie jest opisany protokół pakietu. Warto zauważyć, że tylko bardziej popularniejsze protokoły są opisane swoją nazwą, pozostołe, czyli mniej spotykane są zastąpione TCP/UDP i numerem portu na, którym występują
- Length-jest to długość pakietu
- TTL-tzw czas życia pakietu

W tym oknie również występują przyciski:

- Change interface- jest odpowiedzialny za powrót do poprzedniego okna w celu zmieniania podsłuchiwanego interfejsu
- Pause/Resume- jest odpowiedzialny za zatrzymywania lub wznawianie podsłuchiwania. Szczególnie przydatne, gdy chce się przeanalizować uzyskane pakiety.

Istnieje również opcja żeby bliżej zapoznać się z określonym pakietem. Żeby to zrobić należy 2 razy kliknąć na interesujący pakiet. Wtedy w oddzielnym oknie ukażą się dodatkowe informacje.

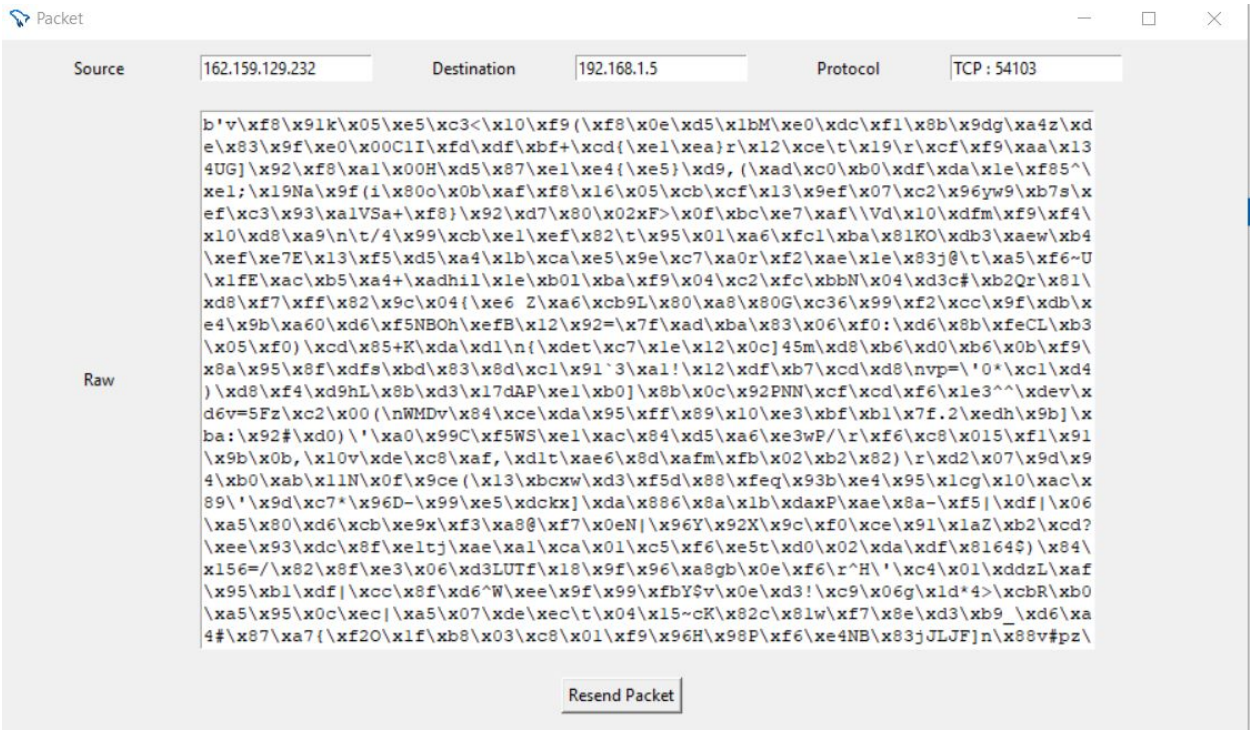
Modyfikacja pakietów



Kolejne okno dostępne w systemie. Jego głównym zadaniem jest wyświetlenie dla użytkownika informacji surowych danych pakietu oraz ich modyfikacji. Mamy w sumie 4 pola:

- Source- adres IP nadawcy pakietu
- Destination- adres IP odbiorcy pakietu

- Protocol- protokół pakietu
- Raw- surowe dane pakietu*(warto pamiętać, że wiele pakietów ma te pole zaszyfrowanie dlatego prawie niemożliwy jest odczyt)*



Żeby zmodyfikować pakiet wystarczy zmienić wybraną opcję w oknie. Żeby wysłać pakiet należy kliknąć przycisk Resend Packet. Następnie warto chwilę poczekać i powrócić do okna z pakietami w celu sprawdzenia czy pakiet został poprawnie odesłany.