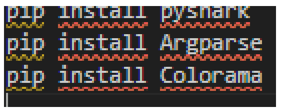
DNS Cache Monitoring and Malicious IP Detection

What is DNS?
The Domain Nam eSystem is an essential part of internet browsing today. Its capabilities help users around the world by translating the website address inputted into your browser into its corresponding IP address. When we enter website addresses we never first think of entering the ip address however, the site name instead. That is what DNS is.

Why should we monitor it?
DNS monitoring is the practice of keeping records and analyzing them from DNS records, performance and availability to ensure that domain resolution works as expected.

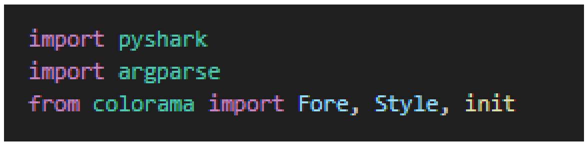
Overview:
For this project we will be creating a simple DNS security tool that monitors and detects malicious attacks such as DDOS/DOS using python.

How to:
Let's install our desired libraries first: For this project we will be using Pyshark, Argparse and Colorama
Pyshark: A python wrapper for Wireshark packet capture tool. This allows us to use our program like Wire shark to monitor network traffic in real-time
Argparse: A module to handle command-line arguments, enabling the customization of the network interface and bytesize threshold
Colorama: This is a library that simplifies colored terminal output which helps us make alerts more noticeable and easier to read



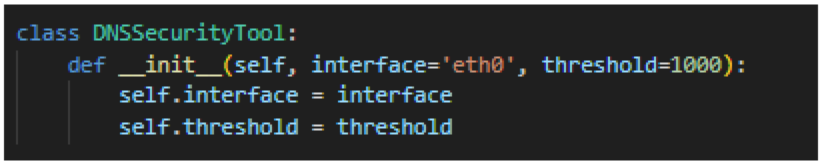
```
pip install pyshark
pip install Argparse
pip install Colorama
```

Main:



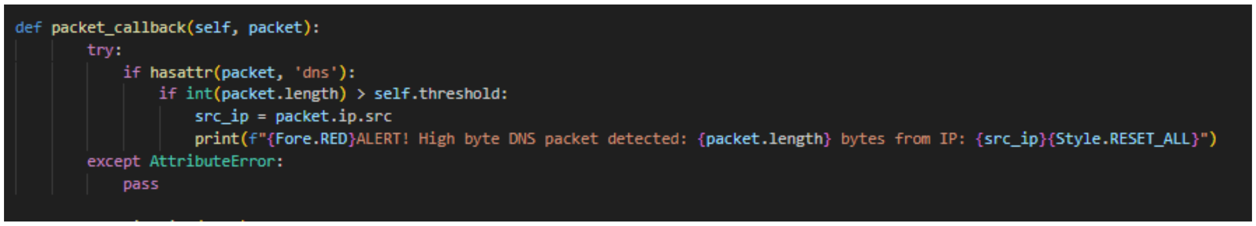
```python
import pyshark
import argparse
from colorama import Fore, Style, init
```

First in out main coding environment, we have to specify all imports such as pyshark, argparse, and colorama
Pyshark is the library used for capturing and analyzing network packets. Having fundamental understanding on how to use wireshark would be helpful for this python program
Additionally, argparse is a module that will help us parse command line arguments, allowing us to customize out tool's behavior without modifying the code
Colorama has a extended library to differentiate colored text output However, using the entire library would be counter intuitive so for this project we will only using an import of Fore, Style, init



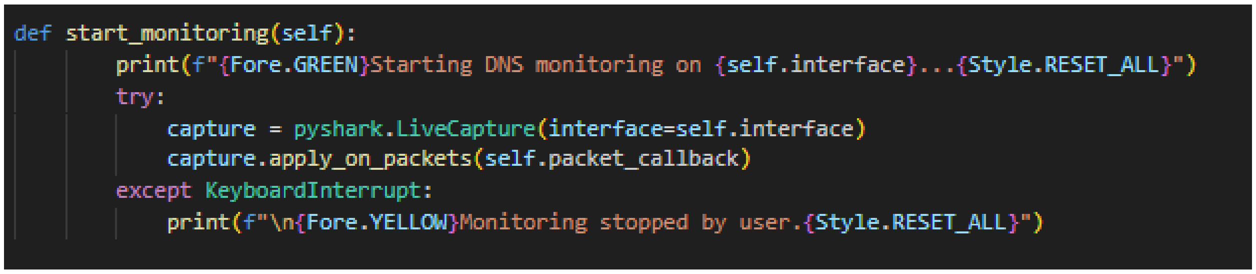
```python
class DNSSecurityTool:
    def __init__(self, interface='eth0', threshold=1000):
        self.interface = interface
        self.threshold = threshold
```

Let us start by calling for class DNSSecurity Tool. In the following code, we have entered init method, which is a constructor that initializes the DNSSecurity Tool class which sets up the network interface. We have set the parameters as eth0 (commonly known as the network interface ethernet) and a byte size threshold of 1000. Self is used as a reference to the current instance of a class, allowing access to its attributes and mehtofs.It will help differentiate between instance variables and local variables



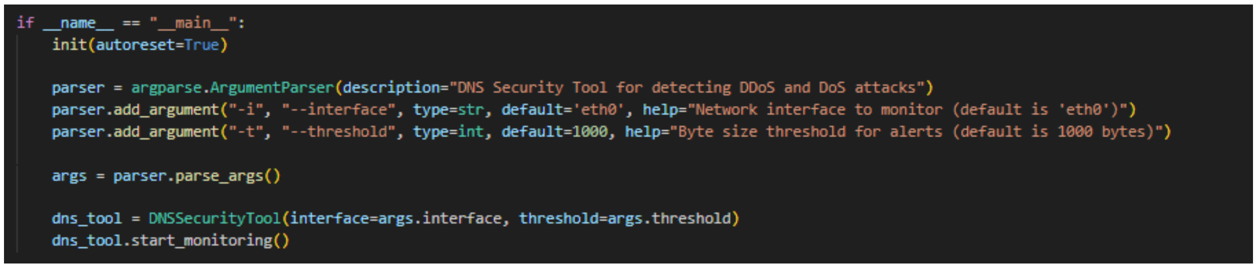
```python
def packet_callback(self, packet):
        try:
            if hasattr(packet, 'dns'):
                if int(packet.length) > self.threshold:
                    src_ip = packet.ip.src
                    print(f"{Fore.RED}ALERT! High byte DNS packet detected: {packet.length} bytes from IP: {src_ip}{Style.RESET_ALL}")
        except AttributeError:
            pass
```

Here we can define packet_callback method. This method is called for each captured packet. It checks if the packet contains DNS data and whether its length exceeds the threshold. The if statement will print out an alert if the threshold before exceeds the limits, outputting the packet length and source IP address in red color.



```python
def start_monitoring(self):
        print(f"{Fore.GREEN}Starting DNS monitoring on {self.interface}...{Style.RESET_ALL}")
        try:
            capture = pyshark.LiveCapture(interface=self.interface)
            capture.apply_on_packets(self.packet_callback)
        except KeyboardInterrupt:
            print(f"\n{Fore.YELLOW}Monitoring stopped by user.{Style.RESET_ALL}")
```

Here we are defining the start_monitoring method. This method starts the live packet capture on the specified network interface. In this case it will eth0 and the capture wil be applied to the packet_callback function to each packet. This monitor will continue to run as long as it is not interrupted by the user, much like Wireshark. In the final line of code, the print function is there to define that monitor has stopped due to user interruption



```python
if __name__ == "__main__":
    init(autoreset=True)

    parser = argparse.ArgumentParser(description="DNS Security Tool for detecting DDoS and DoS attacks")
    parser.add_argument("-i", "--interface", type=str, default='eth0', help="Network interface to monitor (default is 'eth0')")
    parser.add_argument("-t", "--threshold", type=int, default=1000, help="Byte size threshold for alerts (default is 1000 bytes)")

    args = parser.parse_args()

    dns_tool = DNSSecurityTool(interface=args.interface, threshold=args.threshold)
    dns_tool.start_monitoring()
```

Finally, the main execution block will hel run if the script is executed directly. It initialized coloroma to reset colors automatically after each print. It uses the parsing command line argument arpase for the network interface and bytesize threshold. THe DNSSecurityTool will be created as a new instance and it will start monitoring DNS traffic

Conclusion:

This simple DNS security tool serves as the first step towards a full monitoring system. It will monitor DNS traffic and identify unusually large packets which indicate a source of DDos or DoS attack. By customizing the network interface and bytesize threshold we are able to provide flexibility to different network environments for future and further use.