Developing an Algorithm to Identify and Classify Mesh Networks for Enhanced Security

Team **The Red Pill**

Background

The widespread availability of mesh network apps such as Firechat has led to concerns about potential security risks. These apps enable the creation of private networks using frequency bandwidth similar to traditional Bluetooth and Wi-Fi signals, making it challenging to detect and classify them. In particular, terrorists can leverage these apps for covert communication, posing a significant threat to national security. Thus, there is a pressing need for an algorithm that can effectively identify and classify mesh networks to enhance security.

Problem Statement

The primary challenge is to develop an algorithm that can accurately distinguish between normal Wi-Fi signals and mesh network signals generated by apps like Firechat. Current techniques for detecting mesh networks involve analysing transmitted packets and identifying specific parameters that differentiate them from standard Wi-Fi signals. However, these techniques may not always be effective, as it can be challenging to determine which parameters to use for classification.

Solution

This project proposes to develop a new algorithm to address the problem of detecting mesh networks. The proposed algorithm will employ advanced machine learning techniques to analyse captured network traffic and accurately classify it as either a mesh network signal or a standard Wi-Fi signal. We will also experiment with different parameters and analyse their effectiveness in identifying and classifying mesh networks.

Methodology

The project will be conducted in two phases. In the first phase, we will capture network traffic generated by Firechat and other mesh network apps, as well as standard Wi-Fi signals, and analyse them to identify key parameters that differentiate between the two types of signals.
We plan to employ dictionary attacks to crack the Wi-Fi passwords and connect to the network inorder to receive the packets for analysis . A dictionary attack is a type

of cyber attack that involves systematically attempting to guess a password or encryption key by using a large list of words from a pre-existing dictionary of common words, phrases, or character combinations.
In the second phase, we will develop and test the algorithm using machine learning techniques and evaluate its performance based on its ability to accurately classify network traffic as either a mesh network signal or a standard Wi-Fi signal.

Expected Outcomes

The proposed algorithm is expected to provide a reliable and effective solution for detecting and classifying mesh networks. By identifying and blocking covert communication channels used by terrorists and other malicious actors, the algorithm will enhance national security and protect against potential threats to public safety.

Conclusion

The proposed project aims to develop a novel algorithm that can detect and classify mesh networks effectively. The successful implementation of this project will contribute to the development of enhanced security measures against covert communication channels used by terrorists and other malicious actors.

Inorder to simulate the mesh network we use our own test environment using python code similar to below

```
import os
import socket
import threading

# Set the wireless interface to ad-hoc mode
os.system("sudo iwconfig wlan0 mode ad-hoc")

# Set the ESSID (network name) of the ad-hoc network
os.system("sudo iwconfig wlan0 essid my_adhoc_network")

# Set the channel number for the ad-hoc network
os.system("sudo iwconfig wlan0 channel 1")

# Set the IP address for the wireless interface
os.system("sudo ifconfig wlan0 192.168.1.1 netmask 255.255.255.0")
```

```
# Set the IP address and port number of the remote node to chat with
REMOTE_HOST = "192.168.1.2"
REMOTE_PORT = 12345

# Create a UDP socket for sending and receiving messages
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

def receive_messages():
    # Continuously receive messages from the remote node
    while True:
        message, address = sock.recvfrom(1024)
        if address[0] == REMOTE_HOST:
            print(f"Received message: {message.decode('utf-8')}")
        else:
            print(f"Broadcasting message: {message.decode('utf-8')}")
            sock.sendto(message, ("255.255.255.255", REMOTE_PORT))

def send_messages():
    # Continuously send messages to the remote node
    while True:
        message = input("Enter a message: ")
        sock.sendto(message.encode("utf-8"), (REMOTE_HOST,
REMOTE_PORT))

# Start a thread to receive messages
threading.Thread(target=receive_messages, daemon=True).start()

# Start a thread to send messages
threading.Thread(target=send_messages, daemon=False).start()
```