

# IMPLEMENT PACKET SNIFFING USING RADII SOCKETS IN PYTHON.

AIM:

To capture and display packet details ( source, MAC, destination, payload) using Python will stay on windows .

ALGORITHM:

1. Set up the necessary imports from the Scapy library .
2. Define a function to handle the capture packets extracting and printing the relevant information .
3. Call Print a message to indicate that the script is listening for a packet .
4. Call the sniff function from scapy . passing the packet - call back function .
5. The script will capture a single packet or process it using packet - call back function .

CODE:

```

from scapy.all import sniff Ether
def packet_callback(packet):
    if Ether in packet:
        ether: packet[Ether]
        print(f"source MAC: {ether.src}")
        print(f"Destination MAC: {ether.dst}")
        print(f"Payload : {bytes(packet.payload)} - hex({packet.payload})[:100]")
        print("-" * 60)
    
```

```
print ("Hunting for a packet....(press ctrl+c to stop)")  
sniff(prn=packet_callback, count=1)
```

RESULT

Thus a single packet is sniffed & source, destination MA are displayed with payload in windows.

IMPLEMENTATION OF IEEE 802.11

Source MAC: 3C 15: C2 1B: 5B: d1: 80

DESTINATION  
MAC: ff: ff: ff: ff: ff: ff

Payload: 080045000028d4314006401168611608016  
80400

Challenging environment on a road  
- shadowing is present due to buildings

MITIGATION

Set up of stronger processor and go for  
- parallel processing

Set up of robust and efficient  
processing for protocols involving encoding  
- watermarking to tolerate with

fast storage and efficient data base  
- setting is not present in hardware unit

- use most robust fine set and a  
- watermarking based lossless coding with processor

apply a strategy like queue with a  
- lossless coding prior to memory in terms of  
- watermark and

with fine tuning like type of  
- (bitstream) address, - lossless fe  
- lossless coding with watermark

("Fast watermarking", "Fast watermarking", "Fast watermarking")

("Efficient watermarking", "Efficient watermarking", "Efficient watermarking")

("Lossless coding", "Lossless coding", "Lossless coding")

("Fast watermarking", "Fast watermarking", "Fast watermarking")

("Fast watermarking", "Fast watermarking", "Fast watermarking")