Network Packet Sniffing

Sniffing or network packet sniffing is the process of monitoring and capturing all the packets passing through a given network using sniffing tools. It is a form wherein, we can "tap phone wires" and get to know the conversation. It is also called **wiretapping** and can be applied to the computer networks.

There is so much possibility that if a set of enterprise switch ports is open, then one of their employees can sniff the whole traffic of the network. Anyone in the same physical location can plug into the network using Ethernet cable or connect wirelessly to that network and sniff the total traffic.

In other words, Sniffing allows you to see all sorts of traffic, both protected and unprotected. In the right conditions and with the right protocols in place, an attacking party may be able to gather information that can be used for further attacks or to cause other issues for the network or system owner.

How does sniffing work?

A sniffer normally turns the NIC of the system to the promiscuous mode so that it listens to all the data transmitted on its segment.

The promiscuous mode refers to the unique way of Ethernet hardware, in particular, network interface cards (NICs), that allows an NIC to receive all traffic on the network, even if it is not addressed to this NIC. By default, an NIC ignores all traffic that is not addressed to it, which is done by comparing the destination address of the Ethernet packet with the hardware address (MAC) of the device. While this makes perfect sense for networking, non-promiscuous mode makes it difficult to use network monitoring and analysis software for diagnosing connectivity issues or traffic accounting.

A sniffer can continuously monitor all the traffic to a computer through the NIC by decoding the information encapsulated in the data packets.

Types of Sniffing

Sniffing can be either Active or Passive in nature. We will now learn about the different types of sniffing.

Passive Sniffing

In passive sniffing, the traffic is locked but it is not altered in any way. Passive sniffing allows listening only. It works with the Hub devices. On a hub device, the traffic is sent to all the ports. In a network that uses hubs to connect systems, all hosts on the network can see the traffic. Therefore, an attacker can easily capture traffic going through.

The good news is that hubs have almost become obsolete in recent times. Most modern networks use switches. Hence, passive sniffing is no more effective.

Active Sniffing

In active sniffing, the traffic is not only locked and monitored, but it may also be altered in some way as determined by the attack. Active sniffing is used to sniff a switch-based network. It involves injecting address resolution packets (ARP) into a target network to flood on the switch content addressable memory (CAM) table. CAM keeps track of which host is connected to which port.

The Sniffing Effects on Protocols

Protocols such as the **tried and true TCP/IP** were never designed with security in mind. Such protocols do not offer much resistance to potential intruders. Following are the different protocols that lend themselves to easy sniffing −

HTTP

It is used to send information in clear text without any encryption and thus a real target.

SMTP (Simple Mail Transfer Protocol)

SMTP is utilized in the transfer of emails. This protocol is efficient, but it does not include any protection against sniffing.

NNTP (Network News Transfer Protocol)

It is used for all types of communication. A major drawback with this is that data and even passwords are sent over the network as clear text.

POP (Post Office Protocol)

POP is strictly used to receive emails from the servers. This protocol does not include protection against sniffing because it can be trapped.

FTP (File Transfer Protocol)

FTP is used to send and receive files, but it does not offer any security features. All the data is sent as clear text that can be easily sniffed.

IMAP (Internet Message Access Protocol)

IMAP is same as SMTP in its functions, but it is highly vulnerable to sniffing.

Telnet

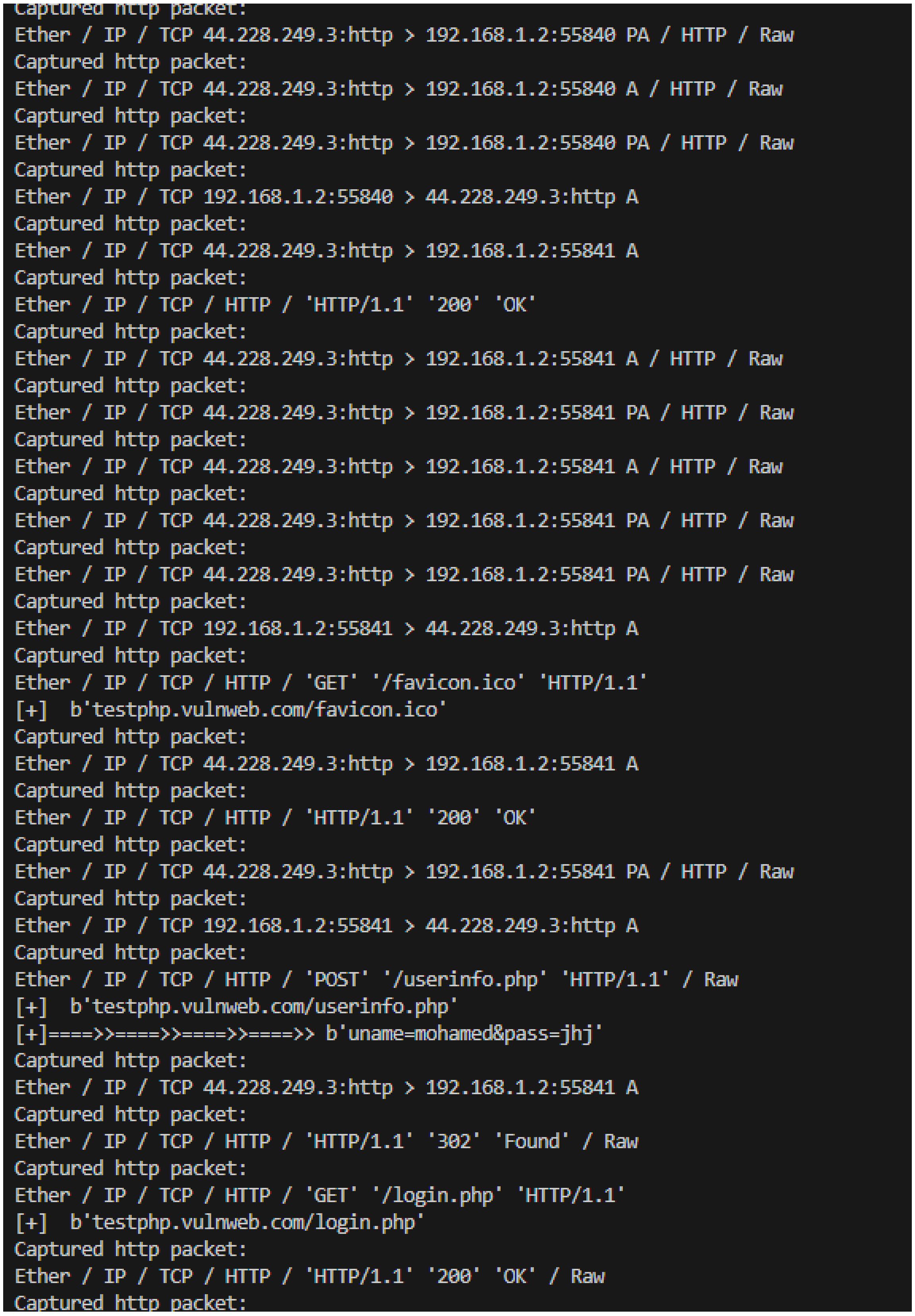
Telnet sends everything (usernames, passwords, keystrokes) over the network as clear text and hence, it can be easily sniffed.

Sniffers are not the dumb utilities that allow you to view only live traffic. If you really want to analyze each packet, save the capture and review it whenever time allows.

Implementation using Python:

```
: import scapy.all as scapy

from scapy.layers import http

# Define the list of protocols you want to capture
protocols = {"http": 80, "smtp": 25, "pop3": 110, "nntp": 119, "ftp": 21,
"telnet": 23, "imap": 143}

def sniff_packets(interface):# the argument of interface is the name of network
card
    print("----------------------------------")
    print("[+] * Sniffer Has Started... * [+]")
    print("----------------------------------")
    try:
                       #interface    #don't store data captured
        scapy.sniff(iface=interface, store=False, prn=process_packet)#prn if find
packets call function process and send the packet to it as argument
    except KeyboardInterrupt:
        print("Sniffing stopped.")

def process_packet(packet):
    if packet.haslayer(scapy.TCP):
        src_port = packet[scapy.TCP].sport
        dst_port = packet[scapy.TCP].dport
        for protocol, port in protocols.items():
            if src_port == port or dst_port == port:
                print(f"Captured {protocol} packet:")
                print(packet.summary())

        if packet.haslayer(http.HTTPRequest):
            print("[+]HTTP====>>====>>====>>====>>",
packet[http.HTTPRequest].Host + packet[http.HTTPRequest].Path )
            if packet.haslayer(scapy.Raw):  #if there get/post request sent from
this host print it
                request= packet[scapy.Raw].load
                print("[+]====>>====>>====>>====>>====>>====>>====>>====>>====>>=
===>>====>>====>>",request)


if __name__ == "__main__":
    interface = "Wi-Fi"  # Change this to your network interface (e.g., "eth0"
for Ethernet)

    print("Sniffing packets for specified protocols...")
    sniff_packets(interface)
```

the output:



```
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55840 PA / HTTP / Raw
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55840 A / HTTP / Raw
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55840 PA / HTTP / Raw
Captured http packet:
Ether / IP / TCP 192.168.1.2:55840 > 44.228.249.3:http A
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 A
Captured http packet:
Ether / IP / TCP / HTTP / 'HTTP/1.1' '200' 'OK'
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 A / HTTP / Raw
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 PA / HTTP / Raw
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 A / HTTP / Raw
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 PA / HTTP / Raw
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 PA / HTTP / Raw
Captured http packet:
Ether / IP / TCP 192.168.1.2:55841 > 44.228.249.3:http A
Captured http packet:
Ether / IP / TCP / HTTP / 'GET' '/favicon.ico' 'HTTP/1.1'
[+]  b'testphp.vulnweb.com/favicon.ico'
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 A
Captured http packet:
Ether / IP / TCP / HTTP / 'HTTP/1.1' '200' 'OK'
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 PA / HTTP / Raw
Captured http packet:
Ether / IP / TCP 192.168.1.2:55841 > 44.228.249.3:http A
Captured http packet:
Ether / IP / TCP / HTTP / 'POST' '/userinfo.php' 'HTTP/1.1' / Raw
[+]  b'testphp.vulnweb.com/userinfo.php'
[+]====>>====>>====>>====>> b'uname=mohamed&pass=jhj'
Captured http packet:
Ether / IP / TCP 44.228.249.3:http > 192.168.1.2:55841 A
Captured http packet:
Ether / IP / TCP / HTTP / 'HTTP/1.1' '302' 'Found' / Raw
Captured http packet:
Ether / IP / TCP / HTTP / 'GET' '/login.php' 'HTTP/1.1'
[+]  b'testphp.vulnweb.com/login.php'
Captured http packet:
Ether / IP / TCP / HTTP / 'HTTP/1.1' '200' 'OK' / Raw
Captured http packet:
```

In the context of the captured packets you provided, the abbreviations "A", "FA", "R", and "SA" represent the flags associated with TCP packets. These flags are part of the TCP header and indicate the state or purpose of the TCP connection. Here's what each flag means:

- **A (Acknowledge)**: This flag indicates that the acknowledgment number field is significant. It's used to acknowledge received data.

- **FA (Finish-Acknowledge)**: This flag indicates the end of data transmission from the sender. It's used to initiate the termination of a connection and acknowledge the receipt of the final segment in a connection termination handshake.

- **R (Reset)**: This flag indicates that the receiver is rejecting the connection attempt. It's used to reset a connection that is in a non-synchronized state.

- **SA (Syn-Acknowledge)**: This flag is used during the TCP three-way handshake to acknowledge the receipt of the SYN packet and to initiate the establishment of a connection.

Here's how these flags are typically used in the TCP connection lifecycle:

1. **SYN**: The sender initiates a connection by sending a SYN packet.
2. **SYN-ACK**: The receiver responds with a SYN-ACK packet to acknowledge the receipt of the SYN packet and to indicate its readiness to establish a connection.
3. **ACK**: The sender acknowledges the SYN-ACK packet, and the connection is established.
4. **Data Transfer**: Data transfer occurs between the sender and receiver.
5. **FIN**: When one party (sender or receiver) wants to close the connection, it sends a FIN packet.
6. **FIN-ACK**: The other party acknowledges the FIN packet.
7. **FIN-ACK (second time)**: The other party also sends a FIN packet to initiate its termination of the connection.
8. **ACK (second time)**: The original sender acknowledges the second FIN packet.
9. **Connection Closed**: The connection is now closed.

These flags are essential for understanding the state of TCP connections and diagnosing network issues. In your captured packets, you can see various combinations of these flags indicating different stages of TCP communication.