Covert Channels

Comp 8505 Assignment 1

Mat Siwoski - A00758640

Introduction/Analysis

The purpose of this assignment was to become familiar with covert channels and to design a covert channel using the TCP/IP protocol suite. Further, Craig Rowland's application "covert_tcp.c" was to be analysed for any shortcomings within his code.

In the application, "covert_tcp.c", Craig Rowland uses three techniques for embedding covert messages within the header:

1.) Bounce Server: Set the ISN to a value of one less than the byte and bounce them off another server than to the original server.

2.) IP Header ID Field: Set the IP header ID field to the data intended for transfer and sending the data as normal. The IP Header ID Field is for reassembly of large fragments of packets.

3.) TCP Initial Sequence Number: Set the number to the data intended for transfer and repeatedly sending SYN packets to a remote listening host.

There are some weaknesses with Craig Rowland's code in "covert_tcp.c".

1.) Network infrastructure: Network infrastructure can manipulate and change the IP ID field. The reason this can happen is that this field is used for large amounts of traffic.

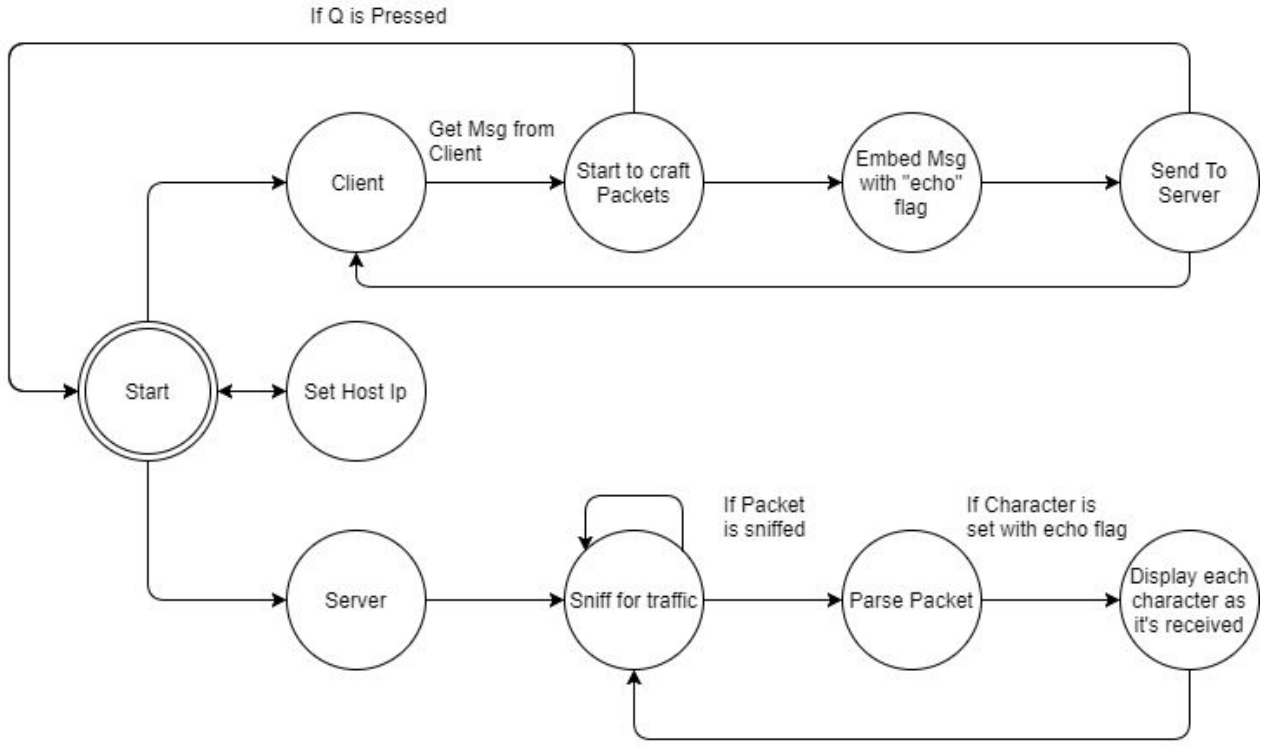
2.) Observant network administrators: By manipulating the TCP initial sequence number, if the network has an observant network administrator, they may be able to notice that the numbers are not necessarily falling into correct sequence. Although this is a possibility, it is still very difficult to notice and may still succeed.

For my assignment, I decided to use the control flags within the TCP Header, specifically to use the **ECHO** flag within the header. By using the Scapy packet manipulation program, I am able to manipulate and insert a character in place of the **ECHO** flag.

Application Requirements

- Your technique for embedding covert data into the headers must be one that is not covered by any of the techniques in the paper.
- You may only use the TCP, UDP, or IP headers for this exercise.
- You are required to show all the data supporting the success (or lack thereof) of your data embedding scheme.

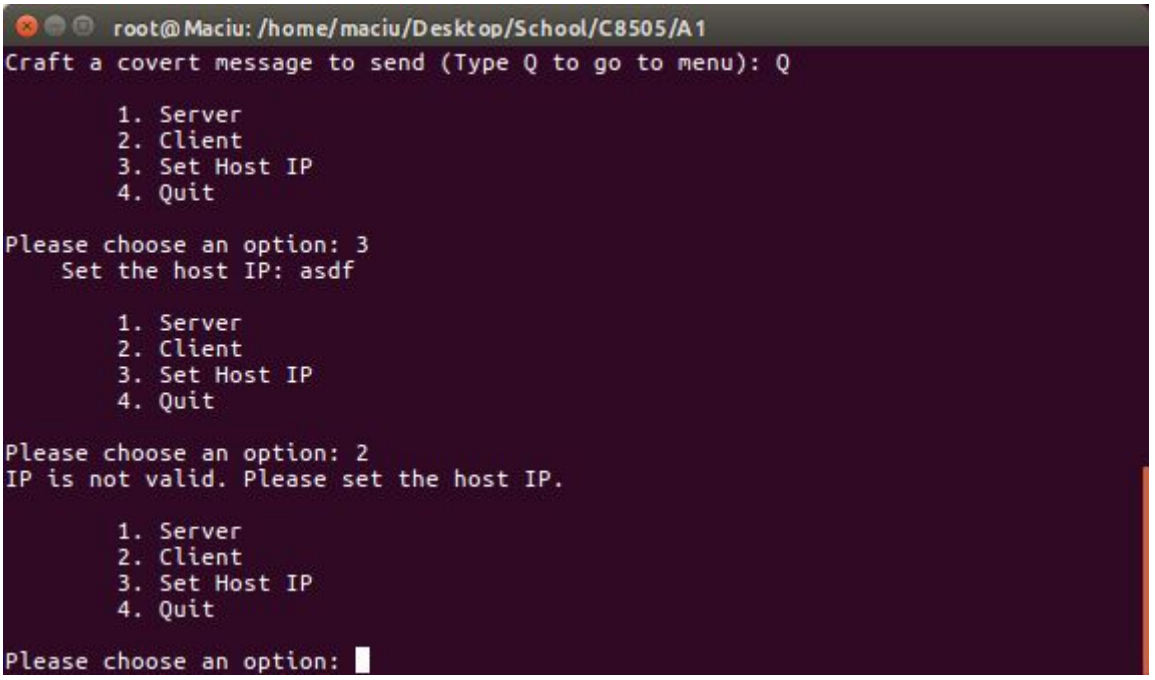
Design


If Q is Pressed
Client
Get Msg from Client
Start to craft Packets
Embed Msg with "echo" flag
Send To Server
Start
Set Host Ip
Server
If Packet is sniffed
Sniff for traffic
If Character is set with echo flag
Parse Packet
Display each character as it's received

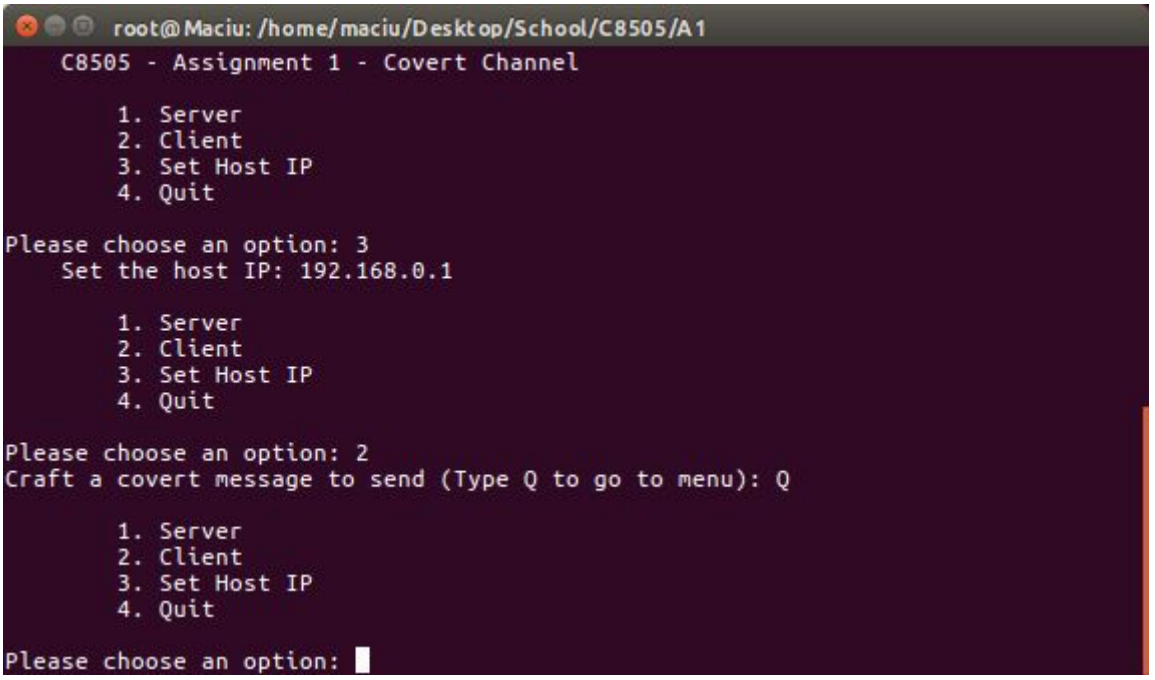
Tests

Here are the results of testing the application.

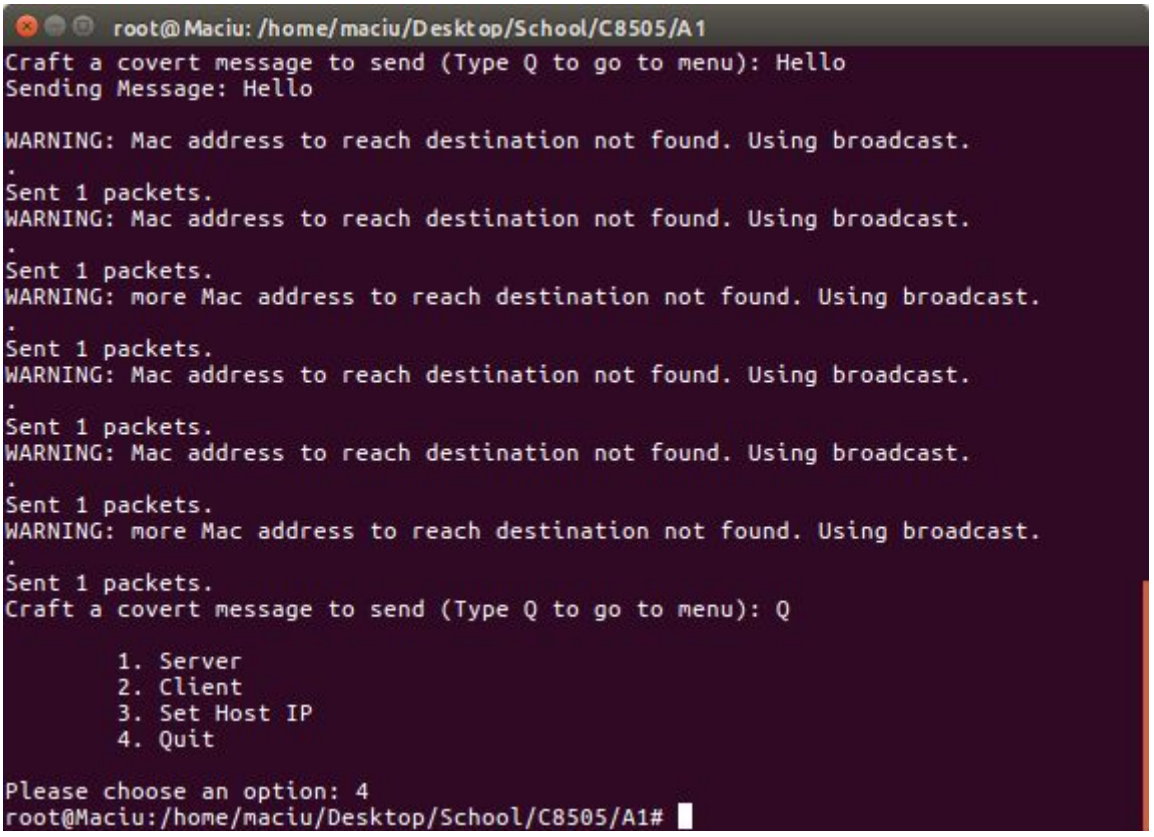
1.) IP Not Valid



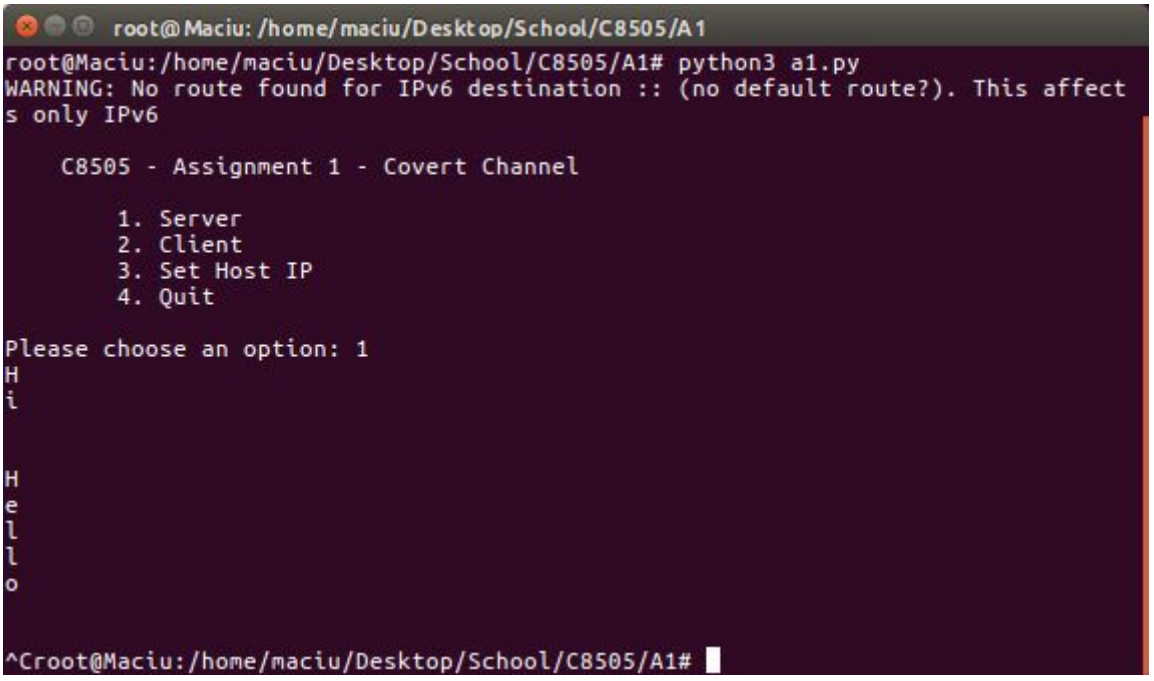
2.) Setting Host IP



3.) Client Sending a message



4.) Server receiving the message



5.) Wireshark Result