Covert Messaging

Assignment 1 for 8505 By Ramzi Chennafi

About this Project

In this project I will be adapting the project created by Craig Rowland. I recreated the code myself as his own was frightening to look at.

The issue I decided to tackle with Rowlands code was creating a tactic of hiding the fact that any sort of connection was taking place between two terminals.

In Rowland's code, he does all communication between two machines. These two machines show a definite link and can defeat the purpose of covert communication. I decided to go over this problem by using SYN floods. In doing these SYN floods I used a random ip from a generated list for each SYN packet. To make them seem more random as the behaviour with SYN flooders is, the source port appears to be randomized. But this is where the actual message data is stored.

By constantly doing SYN floods from different Ips, we are able to completely dissociate the sender from the reciever.

My overall scheme worked well, and it was able to transfer data, albeit – slowly. In the future I would like to work on a better encoding scheme for the data and a dynamically generated list of addresses to further the dissociation from the sender.

Requirements

-Linux
-gcc compiler
-root access (raw sockets cannot be used without root)

Using the program

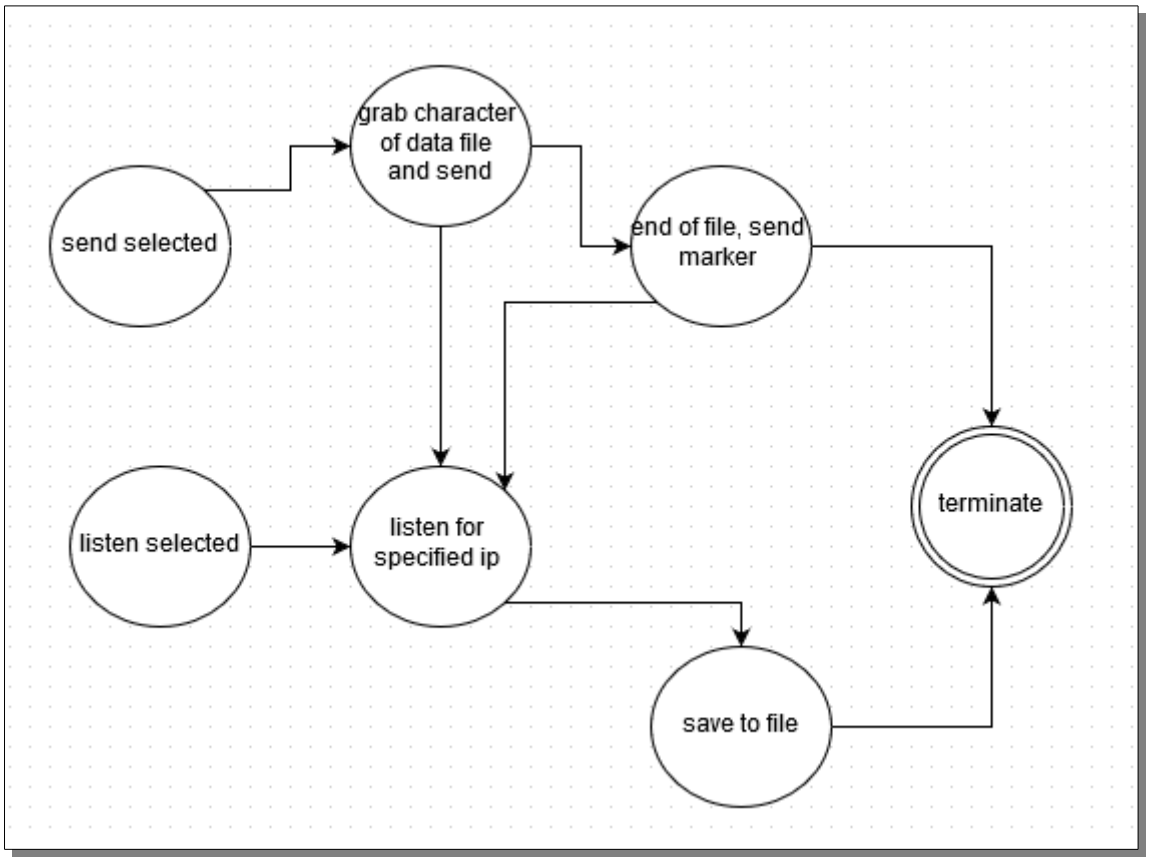
TO send a file type this at execution
"sudo covertsnd send [destination address] [filename]"
TO recieve a file type this at execution
"sudo covertsnd listen [host address]"

The program will terminate once message transfer is completed. The final message is saved to the file dump inthe same directory as the program.

Compiling the program

TO compile the program execute the following on the main directory
gcc *.c -o covertsnd

Design



The overall design of the program is rather simplistic, but it achieves what is done. Whenever a character is sent a new address is chosen from the listing.

Pseudocode

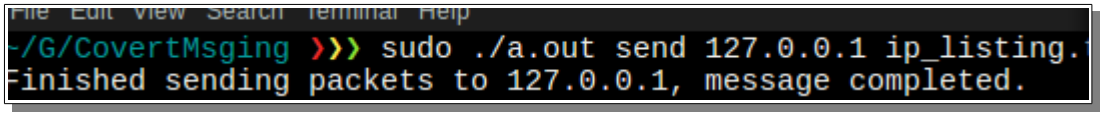
```
int main{
    read in arguments
    read in ip list
    if send is specified
       send_message()
    if recieve is specified
       recieve_message()
}

void recieve_message(){
     open socket and dump file
```

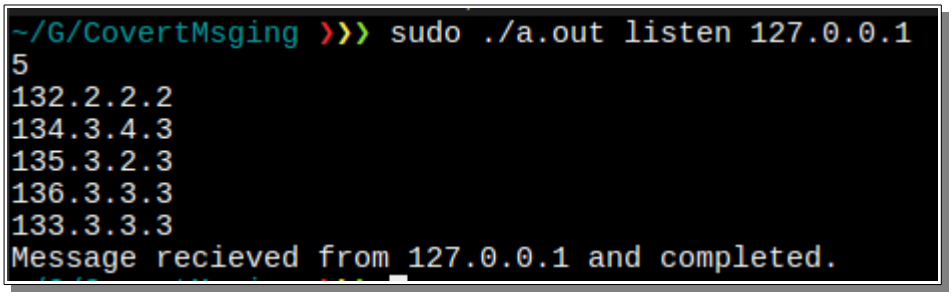
```
    while(){
      recieve data
      check if packet is for us
      if packet for us, process_packet()
        if returns -1 or EOF, the message has ended
        if returns a character, this is part of the message
        if returns a 0, continue till next packet
    }
}

char process_packet(){
    check if packet is
        the tcp protocol
        source address is on the ip listing
        destination address is this machine
                if it is, return the source port character
                else return 0
}

void send_message(){
    open socket and file for reading
    while(NOT end of file){
      send_packet(character)
      sleep for 10 millaseconds -> prevents flood
    }
    send_packet(end of file)
}

void send_packet(){
    craft_packet with a random ip from the listing with the character
    passed to send packet
    sendto the desitination
}

packet craft_packet(){
     create packet
     - ensure it follows the characteristics of a syn packet
     - place the character into the source port
     - set source to the specified random ip address
     - set destination to the specified ip address
     calculate checksum and add to packet
}
```

Testing

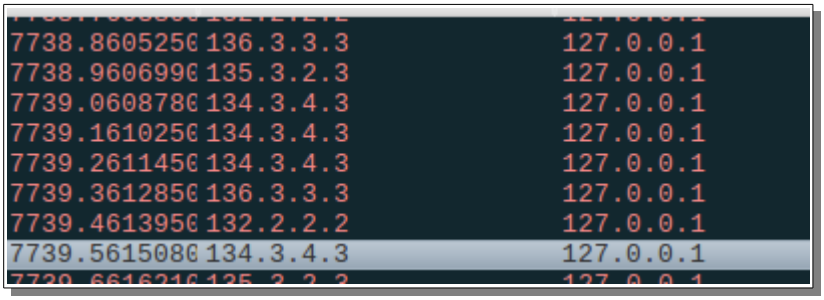
Test No	Name	Desc.	Tools Used	Pass/Fail
1	Data Sent/Recieved	Check if data is sent and received.	Terminal, covertsnd	Pass
2	Packets Sent with varying IPs	Check if the Ips on the packets change.	Wireshark, covertsnd	Pass
3	Data Written to File	Check if file is properly written.	Less, covertsnd	Pass
4	Packets Valid	Check if packets are marked as invalid.	Wireshark, covertsnd	Pass

Test 1 – Data Sent/Recieved





As you can see in the first picture I sent my packet listing file as a covert message. This file was listened for and appeared on the other end. If we look at the IPs used to send it we will also note that the Ips correlate to the output here.



Test 2 – Packets Sent with Varying IPs

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	136.3.3.3	127.0.0.1	TCP	54	13568→80 [SYN] Seq=0 Win=8192 Len=0
2	0.100164000	135.3.2.3	127.0.0.1	TCP	54	2560→80 [SYN] Seq=0 Win=8192 Len=0
3	0.200297000	134.3.4.3	127.0.0.1	TCP	54	12544→80 [SYN] Seq=0 Win=8192 Len=0
4	0.300483000	136.3.3.3	127.0.0.1	TCP	54	13056→80 [SYN] Seq=0 Win=8192 Len=0
5	0.400595000	134.3.4.3	127.0.0.1	TCP	54	12800→80 [SYN] Seq=0 Win=8192 Len=0
6	0.500724000	136.3.3.3	127.0.0.1	TCP	54	11776→80 [SYN] Seq=0 Win=8192 Len=0
7	0.600821000	135.3.2.3	127.0.0.1	TCP	54	12800→80 [SYN] Seq=0 Win=8192 Len=0
8	0.700938000	132.2.2.2	127.0.0.1	TCP	54	11776→80 [SYN] Seq=0 Win=8192 Len=0

As you can see here, I retrieved the wireshark sample of the same transfer. The packets were sent to port 80 and the data was stored within the source port. Only 5 addresses were used in the listing here, that is the reason for the lack of variance in the addresses. This test was a success.

Test 3 – Data Written to File

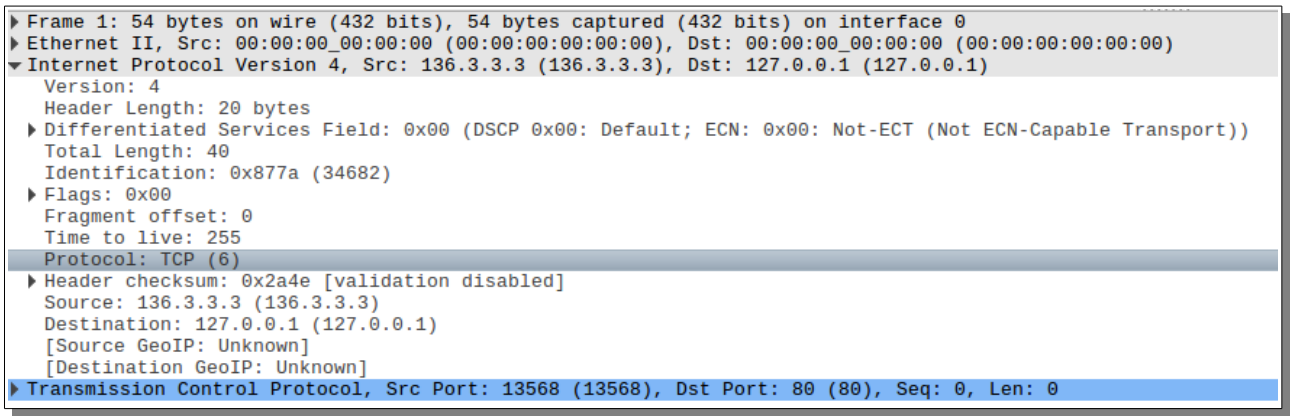
```
~/G/CovertMsging ❯❯❯ sudo ./a.out send 127.0.0.1 ip_listing.
Finished sending packets to 127.0.0.1, message completed.
~/G/CovertMsging ❯❯❯ less dump
5
132.2.2.2
134.3.4.3
135.3.2.3
136.3.3.3
133.3.3.3
```

In this example, we completed the same transfer from before. Afterwards I used less to show the contents of the dump file. Contained within is the same data we saw in the previous example.

```
~/G/CovertMsging ❯❯❯ sudo ./a.out listen 127.0.0.1
5
132.2.2.2
134.3.4.3
135.3.2.3
136.3.3.3
133.3.3.3
Message recieved from 127.0.0.1 and completed.
```

This shows that the test was a success.

Test 4 – Packets Valid



The picture above shows us the results of one packet within the message transmission. As you can see, all values including version, length, protocl and addressing information is correct. In the TCP header it has also been set to the likely values for a SYN packet.

Wireshark has not flagged this packet as malformed. This test was successful.

Conclusion

The project passed all its test and can be called a success.