

Covert Network Channel

Hemal Maniar, Kevin Dsouza, Prateek Pradhan

Northeastern University

CY 6740

Network Security

Abstract

Modern day protections, such as deep packet inspection filters, next-generation firewalls, anomaly-based behavioral intrusion detection systems, and data loss/leak prevention systems, make exfiltration of confidential data from a hacked network node extremely difficult. There are several proposed ways in the literature for establishing covert channels for stealthy data extraction; however, most of them generate network traffic anomalies that could be detected by current defensive network technology. We discuss a novel strategy for building a covert channel using Transmission Control Protocol (TCP) Port Numbers along with its proof-of-concept. Prior to data exfiltration, the data is encrypted and hashed. To maintain the connection compatible with TCP, this approach encrypts our hashed data to be exfiltrated within TCP port numbers. We put our strategy to the test against Snort, an open-source intrusion detection system, and were successful in avoiding detection.

Motivation for the Project

What is the problem you are trying to solve or investigate?

In the field of security, an exploit is deemed successful if it goes unnoticed and makes the least amount of noise possible. The goal of an attacker is to obtain access to an untrusted network or host. After the exploit has been carried out, there is a step known as "post-exploitation."

A stealthy and quiet means of transmitting data to and from a hacked machine is regarded a powerful tool in any attacker's armory. Covert network channels, while restricted in some cases, can be used to disguise data (leaked data) in legitimate network traffic.

Why is it important?

The purpose of such a channel is to hide the fact that it exists at all. This data is frequently transferred in plain sight of potential observers, but it can be practically hard to detect if correctly built. Security through obscurity is illustrated by covert channels that provides a technique for transferring data without the network administrator or other users being aware.

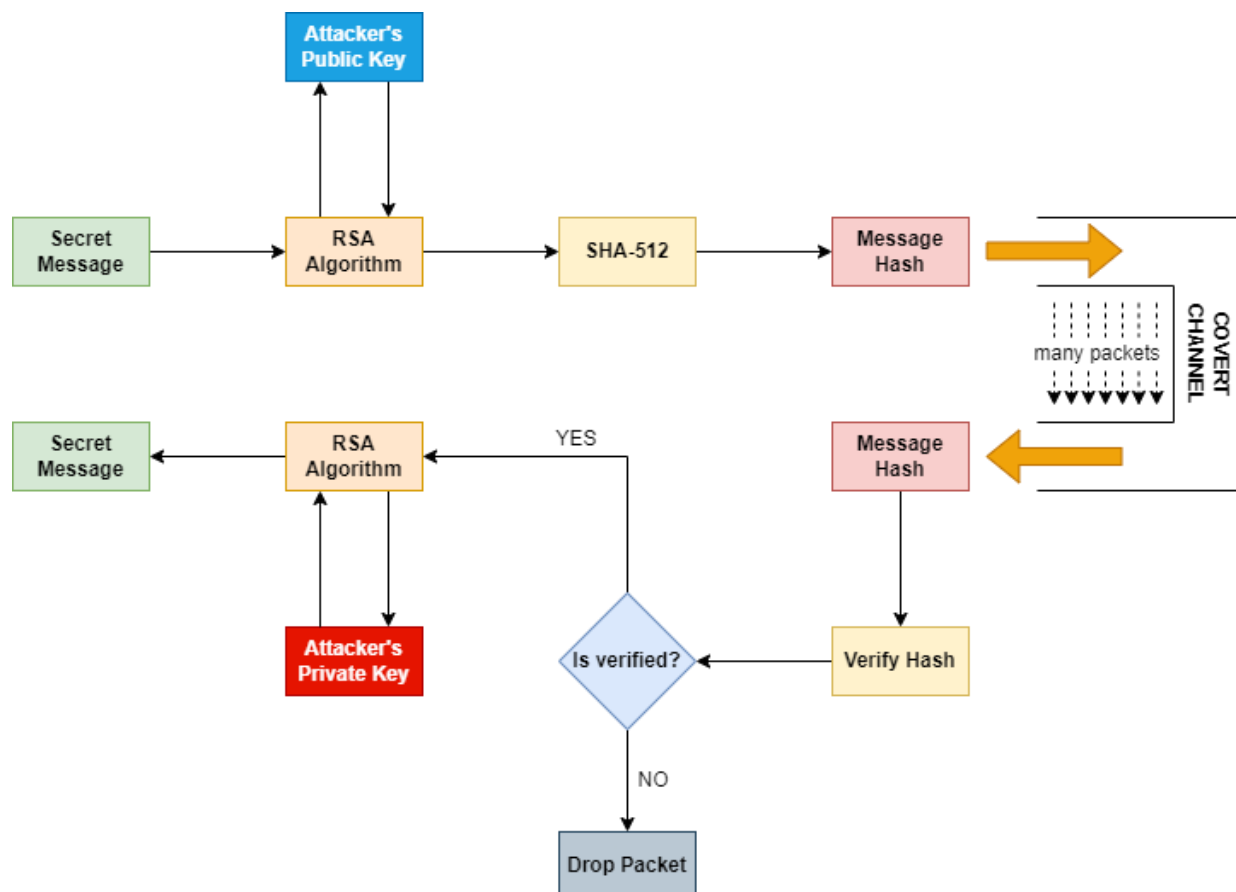
When it comes to computer networks, the term "covert channel" refers to a post-exploitation step in a cyber-attack that allows for the conveyance of information by transferring items across existing information channels or networks and leveraging the structure of the current medium to convey data in small chunks. This virtually eliminates the ability of administrators or users to identify transmission using a covert channel. Data has been stolen from extremely secure systems using covert channels.

Our Project

What are the goals of the project?

The goal of our covert channel deployment methodology is to get data past network administrators without them realizing. This can be accomplished by reducing noise in network traffic, reducing anomaly in network traffic, and eluding intrusion detection systems by masquerading as legitimate network traffic. Our technology also focuses on encrypting and hashing exfiltrated data before sending it in small packets across a covert channel. This is done to ensure the integrity of the data received and to see if any data was manipulated during transmission. We'll use the following methods to put our strategy into action:

1. RSA Encryption
2. SHA-512 Hash



M = Secret Message

E = Encrypted

K_{pub} = Public Key

K_{priv} = Private Key

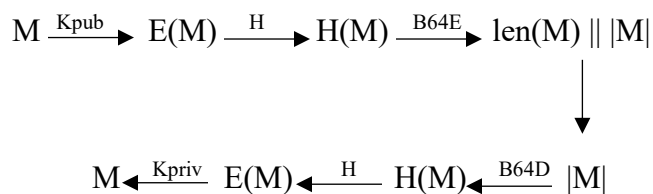
$|M|$ = Message in bytes

\parallel = append

H = SHA512 Hashing algorithm

$B64E$ = Base64 encoding

$B64D$ = Base64 decoding



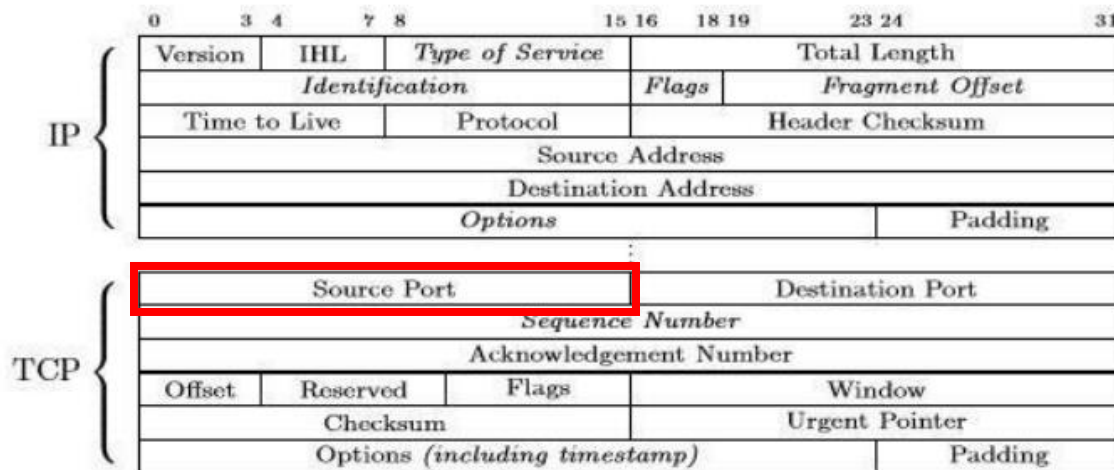
How will achieving those goals help solve the problem you're investigating or discover new, useful information?

Firstly, we implemented a covert channel to test its effectiveness and see if any anomalies could be detected by using Wireshark & Snort IDS. We captured the traffic and analyzed it to check if any data is being leaked to the network administrator which can raise a concern. Our primary goal of implementing a covert channel and enhancing it is to keep it hidden from an observer. By implementing our improved version of covert channels, we were successful in disguising information without raising any alerts in Snort. Not only was the information kept hidden from a network administrator, but we also managed to verify the integrity of received data by verifying its hash. This was done to make sure that the received data was not altered in the process.

Methodology

What methods did you use?

The methods we will be using is sending data using covert channels over TCP. Covert channels are methods to convey information between 2 entities, so that they surpass anomalies and detection surveillance. We will be implementing covert channel and data hiding in the TCP/ IP protocol. The usual TCP/ IP header structure is shown below:



[Figure 1: TCP/IP Header Format]

Initially, covert channels were studied and researched by Rowland. Rowland takes a much more targeted strategy (2010). Rowland devised appropriate encrypting and decrypting procedures by utilizing fields such as the TCP beginning sequence number, the IP identifying field, and the acknowledge field, among others. Rowland demonstrated the existence and manipulation of hidden channels in the TCP/IP protocol suite. Additionally, TCP/IP has a variety of weaknesses such as IP Spoofing, SYN Flooding, Session Hijacking, and Information Hiding.

Did you rely on technologies or data?

Yes, we did rely on the TCP/ IP protocol suite. The TCP/IP suite is designed for simple, open communication between entities, but on the other hand does not provide reliable security or secure channels to its clients. The fundamental features of TCP/ IP to give reliability and flow control opens opportunities, for hiding data using steganographic techniques. There can be many ways and places to hide data in TCP:

OSI Layer Equivalent	TCP/IP Layer	TCP/IP Protocol Examples
Application, session, presentation	Application	NFS, NIS, DNS, LDAP, telnet, ftp, rlogin, rsh, rcp, RIP, RDISC, SNMP, and others
Transport	Transport	TCP, UDP, SCTP
Network	Internet	IPv4, IPv6, ARP, ICMP
Data link	Data link	PPP, IEEE 802.2
Physical	Physical network	Ethernet (IEEE 802.3), Token Ring, RS-232, FDDI, and others

[Figure 2: TCP/IP Stack and Protocols]

Transport Layer:

Maintains reliability of the packets. The source IP and source port number can be spoofed to change it to depict the intended recipient, or the actual data loaded into it. The 3-way handshake done by TCP is an area of interest in this layer. There are 12 fields in the TCP header which are rarely inspected and others that show and have high randomness. For instance, a 32-bit TCP sequence number identifies only the position of the first byte of the whole segment. The source IP and source ports can be manipulated in this too, to deliver wrong information or to deliver some hidden data, like in our case.

On one hand, where most of the commonly known or most used services are assigned a “well-known” or a dedicated port number which is used to manage all the live connections. On the other hand, client ports are not needed to be static and will not always be exactly the same under normal communication. What is done instead is, that the source port is generally a pseudo-random number

chosen from a particular range. Ports that have been chosen like this are called as temporary ports. This is the weakness that gives the client (in this case, team gamma) operating system ability to use any source ports when establishing new connections. There is technically nothing that prevented us from using any port within a 16-bit port range.

Design:

The covert channel works in two phases:

i. Server Initialization (192.168.0.5)

The server will start running on the IP; 192.168.0.5. It will start sniffing for any incoming TCP packets addressed to it. Once a packet is received, it will be converted from decimal to ASCII characters and then decrypted using the decrypt function.

ii. Client Initialization (192.168.0.225)

The client will start running on the IP; 192.168.0.225. It will start sending data hidden in the “sport” field of the TCP header. The destination IP will be supplied by us and will be 192.168.0.5. Once a packet is crafted (ASCII to decimal), it will then be encrypted using the encrypt function.

What metrics did you use to evaluate your work?

Some metrics we used to test our covert channel implementation were packet captures from Wireshark, difficulty to crack encryption and possibility of getting detected using IDS’s. If the covert channel is implemented efficiently:

1. Wireshark captures should not result or gather into a concrete communication information.
2. The encryption used at the client/ server side, should not be easily deciphered.
3. The covert communication should be able to surpass IDS and Firewalls
4. The client (sending the data) should be unaware of the communication.

We have deployed all the above testing methods to test the efficiency of our implementation. The images for the same are given in the results.

How did you collect those measurements?

- i. Start the Server (192.168.0.5).
- ii. Start the Client (192.168.0.225).
- iii. Start Snort IDS (with community rules and covert channel specific rules) in console mode.
- iv. Start Wireshark.
- v. Send Data from the client.
- vi. Check Wireshark and the TCP Stream.
- vii. Check Snort console for any alerts.

Results

What did you accomplish or discover?

Setting up the client and server on separate hosts, we were able to observe the transmission of data and compare the output on the server against data transmitted from the client.

What did your metrics tell you about your efforts?

The aim of our project was to create a covert channel for communication that evades detection and transmits the data successfully from end-to-end. The metrics that we used to evaluate our work conform to these aims and are in accordance with measures that would be employed by a defender. The initial check was to perform a Wireshark capture and compare the transmission against a capture of normal traffic.

```

Enter message to covertly send to server: kevin
Sending message to server: kevin

107
New letter 107
.
Sent 1 packets.
101
New letter 101
.
Sent 1 packets.
118
New letter 118
.
Sent 1 packets.
.
Sent 1 packets.
105
New letter 105
.
Sent 1 packets.
110
New letter 110

.
Sent 1 packets.
10
New letter 10
.
Sent 1 packets.
Message successfully sent to server.
Enter message to covertly send to server: █

```

[Figure 3: Sending unencrypted data over Covert Channel]

```

root@ubuntu:/home/kevin/Downloads/test# python3 server.py
HEY
kevin here
kevin

```

[Figure 4: Receiving unencrypted data over Covert Channel]

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.46	224.0.0.251	IGMPv2	60	Membership Report group 224.0.0.251
3	0.205143851	192.168.0.46	224.0.0.251	MDNS	183	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _compa
4	0.716268055	192.168.0.18	224.0.0.251	MDNS	84	Standard query 0x0e13 PTR 192.168.0.1.in-addr.arpa, "QM" question
5	0.716268137	192.168.0.18	224.0.0.251	MDNS	84	Standard query 0x0e14 PTR 192.168.0.1.in-addr.arpa, "QM" question
6	1.944860141	192.168.0.18	255.255.255.255	UDP	215	34377 → 7437 Len=173
9	3.276938798	192.168.0.46	224.0.0.251	MDNS	183	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _compa
15	5.019037884	192.168.0.18	255.255.255.255	UDP	215	34377 → 7437 Len=173
18	6.801474482	127.0.1.7	192.168.0.225	TCP	54	107 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
19	6.864314988	127.0.1.7	192.168.0.225	TCP	54	101 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
20	6.939316146	127.0.1.17	192.168.0.225	TCP	54	118 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
21	7.006020266	127.0.1.28	192.168.0.225	TCP	54	20 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
22	7.069464215	127.0.1.6	192.168.0.225	TCP	54	105 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
23	7.132798996	127.0.1.7	192.168.0.225	TCP	54	110 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
24	7.191634929	127.0.1.25	192.168.0.225	TCP	54	20 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
25	7.266590280	127.0.1.7	192.168.0.225	TCP	54	10 → 6966 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
26	7.411227066	192.168.0.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
29	8.089495903	192.168.0.18	255.255.255.255	UDP	215	34377 → 7437 Len=173
30	8.413783418	192.168.0.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
33	9.414904990	192.168.0.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
34	9.568208947	192.168.0.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
36	10.416361485	192.168.0.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
37	10.571396815	192.168.0.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
38	10.854180979	192.168.0.18	224.0.0.251	MDNS	84	Standard query 0x0e15 PTR 192.168.0.1.in-addr.arpa, "QM" question
39	10.854181014	192.168.0.18	224.0.0.251	MDNS	84	Standard query 0x0e16 PTR 192.168.0.1.in-addr.arpa, "QM" question
40	11.161324048	192.168.0.18	255.255.255.255	UDP	215	34377 → 7437 Len=173

[Figure 5: Wireshark packet capture for unencrypted transmission]

https://www.rapidtables.com/convert/number/ascii-hex-bin-dec-converter.html

Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB GHDB

Space

☐ 0x/0b prefix

ASCII text

key in

Hex (bytes)

6B 65 76 14 69 6E 14 0A

Binary (bytes)

01101011 01100101 01110110 00010100 01101001 01101110
00010100 00001010

Decimal (bytes)

107 101 118 20 105 110 20 10

Base64

a2V2FGluFAo=

[Figure 6: Decoding the Decimal Values to form ASCII characters]

```

Enter your message to covertly send to server (leave it blank if you want to have it randomized):
Enter message to covertly send to server: hello kevin
Sending message to server: hello kevin

['6', '0', '4', '\n', 'C', 'o', 'A', 'D', 'f', '3', 'u', 'H', 'l', 's', 'x', 'E', 'M', 'a', 'c', 'D', '2', '0', 'l', '1', 'k', 'J', 'x', '2', 'w', 't', 'l', 'B',
'Y', 'l', 'x', '1', 'T', '0', 'J', 'Y', 'D', 'b', '5', 'o', 'Y', 'l', '6', 'q', 'Z', 'I', 'K', 'L', 'U', 'T', 'V', 'm', 'X', 'G', '6', 'J', 'I', '1',
'p', '5', 'l', 'W', 'y', 'r', 't', 'P', '3', 'Z', '8', 'R', 'p', '3', 'X', 'K', 'r', 'c', 'm', 'L', 'p', 'D', 's', 'd', 'f', 'd', 'b', 's', 'N', 'h', 'e',
't', '5', 'C', 'A', 'b', 'w', 'D', 'd', '9', 'f', 'K', 'J', 'Q', 'p', '0', 'd', 'F', 'd', '7', 'C', 'M', 't', 'r', 'B', 'M', 'n', 'X', 'r', 'm', 'f', 'v', 'd',
'v', 'f', 'H', 'C', 'H', 'C', 'H', 'p', 'l', 'c', 'H', 'o', '1', 'n', '4', 'H', 'M', '3', 'T', 's', 'd', 'n', '0', 'W', 'a', 'g', 'q', 'd', 'I',
'X', 't', 'K', 'H', '3', 'U', 'C', 'U', 'M', 'U', '6', 'H', 'X', 'Z', '0', 'Z', 'V', 'C', 'I', 'g', 'I', 'S', 'p', 't', '3', '9', '7', 'w', 'l', 'R', 'B', 'W',
'2', 'v', 'n', 'Z', '4', 'V', 'N', 'l', 'w', '1', 'T', 'W', 'A', 'r', 'Q', 't', 'X', 'P', 'z', '3', 'Z', 'l', 'm', 's', 'N', 'q', 'K', 'x', 'j', 'y', 'w', '9',
'g', 'k', '4', 'q', '4', 'f', 'j', '1', 'h', '6', 'M', '0', 'Q', 'r', 'p', 'n', 'p', 'A', '1', 't', 's', 'Z', 'j', 'r', 'v', 's', 'Y', 't', 'V', '1', '4', 'w', 'Z',
'e', 'V', '3', 'V', '9', 'p', 'T', '0', 'n', 'p', 'T', '2', 'j', 'C', 'T', 'e', 'G', '5', 'x', 'J', '0', 'l', '5', 'z', 'T', 'Y', 'e', 'V', 'Y', 'f', 'H', '1',
'f', 'n', 'A', '9', '0', '4', 'z', 'M', 'h', 'r', 'L', 'D', 'K', 'l', 'Z', 'u', 'C', '8', 't', 'n', 'q', 'a', 'C', 'f', 'E', 'l', 'g', 'L', '1', '0', 'T', 'B', 'I',
'B', 'U', 'G', 'z', 'Y', 'M', 'k', 'z', '9', 'r', 'r', 'x', 'G', 'H', '0', 'p', 'U', 'C', 's', '0', 'a', 'f', '9', '0', 'b', 'n', 'a', '9', 'y', 'z', 'y', 'e',
't', 'c', 'd', 'f', '7', 't', 'r', 'l', 'f', '2', 'F', '8', 'p', 't', '0', '2', 'K', 'C', 'm', 'x', 'v', '0', 'T', '1', '2', 'v', 'Z', 'f', 'y', '1', '6', 'J',
'Z', 'b', 'e', 'G', 'j', 'z', 'X', 'G', 'U', 'd', '8', 'd', 'I', 'f', '5', 'Y', '8', '5', 'e', 'I', '7', 'l', '3', 'B', '2', 'z', 'k', 'N', 'P', 'e', 'I', 'F',
't', 'U', 'q', 'Y', '0', 'G', 'p', '1', '5', 'g', '0', 'V', 'k', 'U', 'G', 'M', 'W', '2', 'e', 'w', 'p', '0', 'd', 'I', '0', 'h', 'A', 'f', 'l', '9', 'M', '7',
'8', 'E', 'k', 'B', 'f', 'M', 'v', '0', 'R', 'M', '0', 'B', 'U', 'E', 'k', 'B', '0', '0', '1', 'Y', '0', 'K', 'Y', '0', 'd', 'a', '4', 'z', 'k', 'X', 'X', 'f',
'X', 'U', 'X', 'c', '4', '9', 'C', 'k', 'b', 'V', 'f', 'W', '5', 'b', '3', 'P', 'h', 'C', 'g', '3', 'K', 'M', 'G', 'J', 'C', '3', 'a', '5', 'b', 'U', 'W', 't',
'S', 'p', 'r', 't', 'b', 'P', 'b', '7', 'M', '0', 'l', 'f', 'g', 'z', 'U', 'U', 'u', 'N', 'V', '2', 'X', 'q', 'b', 'N', 'q', '0', 'C', 'K']
Message successfully sent to server.

```

```

Enter message to covertly send to server: bot is alive
Sending message to server: bot is alive

['6', '0', '4', '\n', 'C', 'o', 'A', 'D', 'f', '3', 'u', 'H', 'l', 's', 'x', 'E', 'M', 'a', 'c', 'D', '2', '0', 'l', '1', 'k', 'J', 'x', '2', 'w', 't', 'l', 'B',
'Y', 'l', 'x', '1', 'T', '0', 'J', 'Y', 'D', 'b', '5', 'o', 'Y', 'l', '6', 'q', 'Z', 'I', 'K', 'L', 'U', 'T', 'V', 'm', 'X', 'G', '6', 'J', 'I', '1',
'p', '5', 'l', 'W', 'y', 'r', 't', 'P', '3', 'Z', '8', 'R', 'p', '3', 'X', 'K', 'r', 'c', 'm', 'L', 'p', 'D', 's', 'd', 'f', 'd', 'b', 's', 'N', 'h', 'e',
't', '5', 'C', 'A', 'b', 'w', 'D', 'd', '9', 'f', 'K', 'J', 'Q', 'p', '0', 'd', 'F', 'd', '7', 'C', 'M', 't', 'r', 'B', 'M', 'n', 'X', 'r', 'm', 'f', 'v', 'd',
'v', 'f', 'H', 'C', 'H', 'C', 'H', 'p', 'l', 'c', 'H', 'o', '1', 'n', '4', 'H', 'M', '3', 'T', 's', 'd', 'n', '0', 'W', 'a', 'g', 'q', 'd', 'I',
'X', 't', 'K', 'H', '3', 'U', 'C', 'U', 'M', 'U', '6', 'H', 'X', 'Z', '0', 'Z', 'V', 'C', 'I', 'g', 'I', 'S', 'p', 't', '3', '9', '7', 'w', 'l', 'R', 'B', 'W',
'2', 'v', 'n', 'Z', '4', 'V', 'N', 'l', 'w', '1', 'T', 'W', 'A', 'r', 'Q', 't', 'X', 'P', 'z', '3', 'Z', 'l', 'm', 's', 'N', 'q', 'K', 'x', 'j', 'y', 'w', '9',
'g', 'k', '4', 'q', '4', 'f', 'j', '1', 'h', '6', 'M', '0', 'Q', 'r', 'p', 'n', 'p', 'A', '1', 't', 's', 'Z', 'j', 'r', 'v', 's', 'Y', 't', 'V', '1', '4', 'w', 'Z',
'e', 'V', '3', 'V', '9', 'p', 'T', '0', 'n', 'p', 'T', '2', 'j', 'C', 'T', 'e', 'G', '5', 'x', 'J', '0', 'l', '5', 'z', 'T', 'Y', 'e', 'V', 'Y', 'f', 'H', '1',
'f', 'n', 'A', '9', '0', '4', 'z', 'M', 'h', 'r', 'L', 'D', 'K', 'l', 'Z', 'u', 'C', '8', 't', 'n', 'q', 'a', 'C', 'f', 'E', 'l', 'g', 'L', '1', '0', 'T', 'B', 'I',
'B', 'U', 'G', 'z', 'Y', 'M', 'k', 'z', '9', 'r', 'r', 'x', 'G', 'H', '0', 'p', 'U', 'C', 's', '0', 'a', 'f', '9', '0', 'b', 'n', 'a', '9', 'y', 'z', 'y', 'e',
't', 'c', 'd', 'f', '7', 't', 'r', 'l', 'f', '2', 'F', '8', 'p', 't', '0', '2', 'K', 'C', 'm', 'x', 'v', '0', 'T', '1', '2', 'v', 'Z', 'f', 'y', '1', '6', 'J',
'Z', 'b', 'e', 'G', 'j', 'z', 'X', 'G', 'U', 'd', '8', 'd', 'I', 'f', '5', 'Y', '8', '5', 'e', 'I', '7', 'l', '3', 'B', '2', 'z', 'k', 'N', 'P', 'e', 'I', 'F',
't', 'U', 'q', 'Y', '0', 'G', 'p', '1', '5', 'g', '0', 'V', 'k', 'U', 'G', 'M', 'W', '2', 'e', 'w', 'p', '0', 'd', 'I', '0', 'h', 'A', 'f', 'l', '9', 'M', '7',
'8', 'E', 'k', 'B', 'f', 'M', 'v', '0', 'R', 'M', '0', 'B', 'U', 'E', 'k', 'B', '0', '0', '1', 'Y', '0', 'K', 'Y', '0', 'd', 'a', '4', 'z', 'k', 'X', 'X', 'f',
'X', 'U', 'X', 'c', '4', '9', 'C', 'k', 'b', 'V', 'f', 'W', '5', 'b', '3', 'P', 'h', 'C', 'g', '3', 'K', 'M', 'G', 'J', 'C', '3', 'a', '5', 'b', 'U', 'W', 't',
'S', 'p', 'r', 't', 'b', 'P', 'b', '7', 'M', '0', 'l', 'f', 'g', 'z', 'U', 'U', 'u', 'N', 'V', '2', 'X', 'q', 'b', 'N', 'q', '0', 'C', 'K']
Message successfully sent to server.

```

[Figure 7: Sending Encrypted Data over the Covert Channel]

```

(root@kali) - [~/Desktop/FInal_NS]
# python3 server.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
WARNING: can't import layer ipsec: cannot import name 'gcd' from 'fractions' (/usr/lib/python3.9/fractions.py)
Received Message is:
hello kevin

Received Message is:
bot is alive

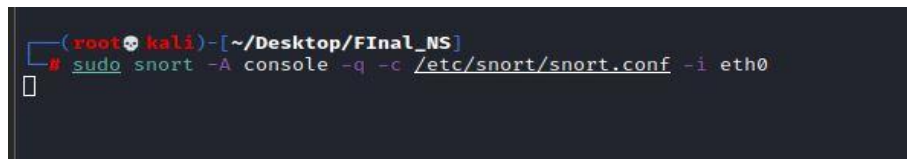
```

[Figure 8: Receiving Encrypted data and decrypting it over Covert Channel]

2697	117.288472405	127.0.1.29	192.168.0.5	TCP	60 88 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2698	117.325160400	127.0.1.8	192.168.0.5	TCP	60 68 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2699	117.353270471	192.168.0.18	255.255.255.255	UDP	215 34377 → 7437 Len=173
2701	117.368820526	127.0.1.28	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2702	117.409028096	127.0.1.19	192.168.0.5	TCP	60 [TCP Retransmission] 115 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2703	117.448901005	127.0.1.19	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2704	117.448906518	127.0.1.27	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2705	117.540110508	127.0.1.15	192.168.0.5	TCP	60 [TCP Retransmission] 117 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2706	117.582769105	127.0.1.27	192.168.0.5	TCP	60 121 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2707	117.629537325	127.0.1.20	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2708	117.672383334	127.0.1.29	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2709	117.712384425	127.0.1.7	192.168.0.5	TCP	60 114 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2710	117.752572405	127.0.1.5	192.168.0.5	TCP	60 [TCP Retransmission] 69 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2711	117.787570976	127.0.1.29	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2712	117.823946302	127.0.1.13	192.168.0.5	TCP	60 [TCP Retransmission] 121 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2713	117.864633593	127.0.1.18	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2714	117.908273915	127.0.1.12	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2715	117.943637480	127.0.1.28	192.168.0.5	TCP	60 98 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2716	117.975725087	127.0.1.24	192.168.0.5	TCP	60 [TCP Retransmission] 70 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2717	118.011619517	127.0.1.6	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2718	118.044643032	127.0.1.13	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2719	118.083623378	127.0.1.18	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2720	118.116331290	127.0.1.20	192.168.0.5	TCP	60 117 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2721	118.163508503	127.0.1.10	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2722	118.195767097	127.0.1.25	192.168.0.5	TCP	60 [TCP Retransmission] 52 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2723	118.228083523	127.0.1.25	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2724	118.264561518	127.0.1.12	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0

[Figure 9: Wireshark Packet Capture for encrypted transmission (nothing concrete shown)]

Analysis of the comparison reveals that Wireshark does not highlight any errors with the captured packets and the traffic flow is quite like normal traffic as well. To ensure that our covert channel also avoids automated detection, we carried out repeated communication between the client and server against an Intrusion Detection System. This served as the second metric to measure our implementation and was completed using Snort. The IDS instance was initiated with community rules, and it did not identify any transmission as malicious. The output from Snort is shown below for reference.



```
(root@kali)~[~/Desktop/FInal_NS]
# sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
```

[Figure 10: SNORT shows no alerts for covert transmission]

2569	112.928399432	127.0.1.10	192.168.0.5	TCP	60 56 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2572	112.969360659	127.0.1.8	192.168.0.5	TCP	60 90 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2573	113.008326635	127.0.1.26	192.168.0.5	TCP	60 117 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2574	113.048398968	127.0.1.9	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2575	113.088727059	127.0.1.23	192.168.0.5	TCP	60 104 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2576	113.135245720	127.0.1.29	192.168.0.5	TCP	60 [TCP Retransmission] 48 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2577	113.176205590	127.0.1.7	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2578	113.212132880	127.0.1.6	192.168.0.5	TCP	60 [TCP Retransmission] 20 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
2579	113.243179708	127.0.1.15	192.168.0.5	TCP	60 116 → 1669 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0

[Figure 11: Wireshark Packet Capture for encrypted transmission]

What obstacles did you encounter, and how could you have overcome those obstacles?

The main obstacle that we faced in the implementation of our project was avoiding detection and choosing a suitable payload delivery mechanism. Analysis of the network traffic using Wireshark and validating the transmission against the Intrusion Detection System was helpful in finalizing our approach.

Currently, our covert channel relies on the use of TCP as the primary delivery mechanism. We analyzed the TCP protocol and successfully designed a program that was able to manipulate and

stealthily transmit packets from the client to the server one byte at a time. The data sent over the network was encrypted and hashed to prevent tampering and ensure confidentiality. However, this implementation can also be modified to include different methods of delivery such as DNS requests, HTTP and HTTPS messages, etc. The next challenge would be to let the client and server chose the delivery model depending on the baseline of the traffic captured from the network. Setting up a sniffer before initiating transmission would allow the client to determine which network protocol to use and proceed accordingly. Similarly, the delay between packets could be determined based on the network traffic as well.

References

1. <https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2017/understanding-covert-channels-of-communication>
2. <https://medium.com/insa-tc/covert-channels-in-computer-networks-26a33fd911b2>
3. <http://web.mit.edu/ha22286/www/papers/HST10.pdf>
4. https://link.springer.com/chapter/10.1007/978-3-319-06320-1_19
5. <https://github.com/cdpXe/Network-Covert-Channels-A-University-level-Course>