Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 1810–1818

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Network Steganography using the Overflow Field of Timestamp Option in an IPv4 Packet

Punam Bedi[a], Arti Dua[b]*

[a,b]*Department of Computer Science, University of Delhi, Delhi - 110007, India.*

Abstract

Steganography is a technique of hiding secret data inside a cover. The most popularly used cover media include images, videos, audios, documents and network protocols. Network Steganography is a technique that uses common network protocols (the header field, the payload field or both) to hide a secret message. TCP/IP protocol suite has been a potential target for network steganography from the very beginning. It has a lot of possibilities for creation of hidden channels that can be used to communicate covertly. In this paper, we propose a technique that creates a covert channel using the Overflow field of Timestamp option of Internet Protocol, version 4 over a Local Area Network. This technique implements a storage based network steganography that uses the timestamp option which is used for debugging and measurement over the networks. In this technique, we use legitimate values in the Overflow field which makes it difficult to detect the possibility of covert communication.

Keywords: Network Steganography; TCP/IP; covert channel; IPv4; Timestamp Option

* Corresponding author.
E-mail address: arti.batra@gmail.com

10.1016/j.procs.2020.04.194

1. Introduction

Information security is the need of every industry today. No organization would want their data to be hacked or intercepted by anyone. One way to secure the data over the networks is through information hiding using steganography. Steganography is the art and science of communicating in a way, which hides the existence of the communication [17]. Steganography takes help of a cover media to hide a secret message. This media could be an image, a video, an audio, a document or a network protocol. Steganography using images, audio and video has been a favorite area of researchers since past two decades. However, Network Steganography is a recent emerging field in the area of research. Other Steganography techniques require extra bandwidth for sending the cover media with hidden data. Whereas with Network Steganography, it is possible to use already existing Protocol Data Units (PDUs) as cover with modifications in redundant fields of the respective PDU. Many commonly used protocols are being proposed for implementing Network Steganography. This hiding of secret information in network protocols' payload or header or both, can be achieved through Network Steganography. Network Steganography offers a good bandwidth for secret data communication. This is because one can create new data packets to carry secret information or modify the already existing data packets to carry covert data. In this paper, we create a hidden channel using the timestamp option of Internet Protocol version 4 (IPv4) which is an optional field of an IPv4 packet. To implement the hidden communication, we make use of User Datagram Protocol (UDP) which is an unreliable transport layer protocol. The UDP segment uses underlying Internet Protocol at the network layer to facilitate the transmission. This is where we create a covert channel using the Overflow field of the timestamp option in the IPv4 header. The scheme is experimented and proposed over a Local Area Network (LAN). The remaining paper is organized as follows: Section 2 and 3 describe the background and related work respectively in the field of Network Steganography. Section 4 elaborates the architecture of the proposed system. Section 5 explains the actual experiment and its setup. Sections 6 and 7 summarize the result and conclusion respectively.

2. Background

2.1. Network Steganography

The term Network Steganography was first used by Krzysztof Szczypiorsky in the year 2003[1]. The main idea of Network Steganography is to hide data in network protocols' header field or payload field or both. Network Steganography is implemented by creating covert channels as means of communication between a covert sender and a covert receiver. These covert channels can be classified into three classes [20]:

- Storage Based Covert channels: These channels carry hidden data in the storage part of Protocol Data unit (PDU). This channel can be created via a redundant field in the header, or a small portion of payload part of the PDU, or both together.
- Timing Based Covert Channels: These covert channels are created on the basis of sequence numbers or timings or delays in the PDUs.
- Hybrid Covert Channels: These covert channels make use of a combination of storage and timing channels together to support covert communication.

The efficiency of a Network Steganography technique can be judged by analyzing the following characteristics [18]:

- Capacity: It is the amount of covert information that a PDU can carry.
- Robustness: Robustness depicts the conformity of the technique to an error free condition where the secret should be resistant to change or failures.
- Imperceptibility: It is an important feature that marks the undetectability of a covert message.

2.2. Local Area Networks

A Local Area Network (LAN) is a collection of a number of devices such as computers, laptops, printers, smart phones etc., connected within a small area like a room, a building or a campus. All the devices in a LAN are connected either through an Ethernet cable (for wired connections) or through some wireless media. In order to

connect this LAN to the Internet, a router is needed which is generally connected to a telephone line to facilitate data transmissions over longer distances as shown in Figure 1.

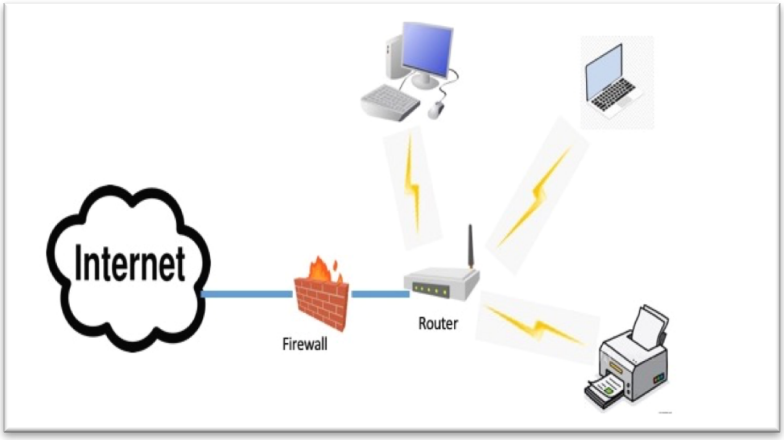

Fig. 1. Local Area Network Connected to the Internet through a Router.

2.3. Internet Protocol version 4

The Internet Protocol version 4 forms the backbone of the Internet. This protocol operates at the network layer which is responsible for taking important decisions like routing of the Internet packets and managing logical addresses over the networks. The header [2] of the Internet Protocol (IP) is given in Figure 2.

Version (4 Bits)	IHL (4 Bits)	Service Type (8 Bits)	Total Length (16 Bits)	
Identification (16 Bits)			Flags (3 Bits)	Fragment Offset (13 Bits)
Time to Live (8 Bits)		Protocol (8 Bits)	Header Checksum (16 Bits)	
Source IP Address (32 Bits)				
Destination IP Address (32 Bits)				
Options (Variable: 0 – 320 Bits)				Padding (Variable)

Fig. 2: Structure of the header of Internet Protocol version 4.

The Version field is the first field of the IP Header. It records the version number of the Internet Protocol being used. The Internet Header Length (IHL) field stores the length of the IP Header (number of 32-bits words) which could be a maximum of 15 as the size of header length field is 4 bits which can hold a maximum value of 15. The next field which is of size 8 bits was earlier known as Type of Service (ToS) field. The first three bits were precedence bits that decided the priority of a packet. The next 4 bits represented high and low values for Delay, Throughput, Reliability and Cost. The next one bit was reserved for future use. The ToS field has now been replaced by Differentiated Code Point (DSCP) which occupies 6 bits and ECN (Explicit Congestion Notification) field which occupies 2 bits. Applications like Voice over IP (VoIP) make use of DSCP field for real time data transmissions and ECN is used for end to end network congestion notification if supported by both the end points [3]. The Total Length field gives the size of IP datagram including the header. It can have a maximum value of 65535 bytes. The

Identification field uniquely identifies an IP datagram. This field is mostly used for fragmentation and reassembly of IPv4 packets. Next 3 bits are Flag bits which are used to identify and control fragments. The first bit is reserved, the second bit is the Don't Fragment (DF) bit which when set to 1, informs the intermediate devices not to fragment the IP packet. The next bit is More Fragment (MF) bit. This bit when set to 1, indicates that this packet is a fragment of much larger packet with other fragments following this one. When set to 0, it indicates that the current datagram is either the last fragment of all the fragments or it is the only fragment. Fragment offset field, tells the offset of this particular fragment relative to the beginning of the original non-fragmented datagram in eight byte blocks. The Time to Live field tells for how long can the packet be alive in a network. It is generally set to the number of hops a packet can travel before getting discarded. The Protocol field tells which protocol at transport layer requested the service of Internet Protocol. The Header checksum which is a 16-bit long field is used to check for the integrity of IP header. A checksum of all the fields (including header checksum field containing all zeros) is computed and stored in this field at the sender side. This is recomputed at the receiver's side and matched with the stored value to check the integrity of IP header. The Source IP Address and Destination IP Address hold the 32 bits long IP address of the sender and the receiver respectively. The last field is Options field which is optional. User may or may not put any data in the Options part of IP header. Some of the possible options include security, loose source routing, strict source routing, timestamp and record route as shown in Table 2 [4]. If any option is present in the data, the first byte of the Option field has a format as depicted in Figure 3. Copy Flag depicts whether this option is copied into all fragments (depicted by value 1) or not (depicted by value 0). The option class is specified using the next two bits as depicted in Table 1. The option number is a 5-bit field. The valid values of option number together with option class, as given in Table 2, specifies an option.

Table 1: Option Class

Option Class	Specification
00 (0)	Control
01 (1)	Reserved for Future Use
10 (2)	Debugging and Measurement
11 (3)	Reserved for Future Use

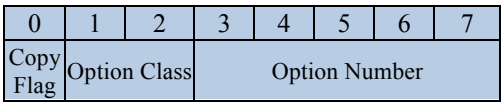

Fig. 3. Option Type Octet

Table 2: Options Specification

Option Class	Option Number	Length	Option Specification
0	0	Nil	End of option list
0	1	Nil	No Operation
0	2	11	Security
0	3	Not Fixed	Loose Source Routing
0	9	Not Fixed	Strict Source Routing
0	7	Not Fixed	Record-route
0	8	4	Stream id
2	4	Not Fixed	Time Stamp

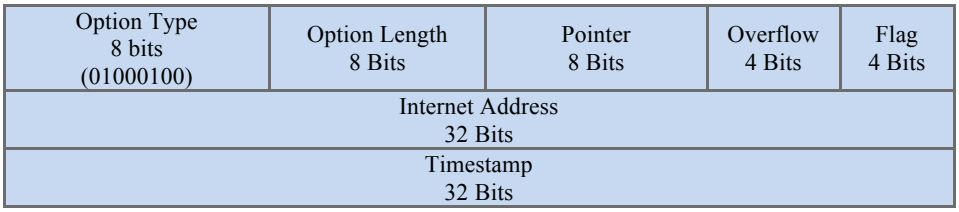

Fig. 4. Structure of IP Timestamp Option

2.4. Structure of IP Timestamp option

The structure of the Timestamp option is shown in Figure 4[8]. The Option Type value for Timestamp option is 68 in decimal (copy flag = 0, option class = 2 and option number = 4). Option Length specifies the number of octets used by the current option including the type, length, pointer, overflow and flag fields. The pointer field is the number of octets beginning from this timestamp option to the end of timestamps plus one. That is, it is the offset of the beginning of next timestamp. The minimum value of this pointer field is 5 and the maximum is 40. The Overflow field is initially zero and is incremented each time a router is unable to add a timestamp to the option field due to lack of space. The flag field has 3 valid interpretations:

- Value 0: Only Timestamps are stored in consecutive 32 bit words
- Value 1: Each registering entity adds its IP address first followed by the Timestamp.
- Value 3: The Internet address field in this case is pre-specified and only those entities enter the timestamps whose IP address match the mentioned address.

3. Related Work

Steganography using images, audio and video has been a favorite area of researchers since the last two decades. Network Steganography is a recent emerging field in the area of research. Many commonly used protocols are being suggested for implementing Network Steganography. In this section, we discuss previous steganography work related to Internet Protocol version 4 only. As per our knowledge, very less work has been done in last few years in implementing steganography in Internet Protocol version 4. Rowland in [13] suggested the use of IP identification field which is normally used for identifying the fragments of an IPv4 packet. In [15] Ahsan and Kundur exploited the redundancy in Fragmentation Strategy of an IPv4 datagram. If a datagram is unfragmented (lesser than the size of maximum transferable unit), there is no use of fields like fragment offset and flags like Don't Fragment and More Fragment. Ahsan and Kundur made use of DF flag to transfer a bit '0' or '1'. Handel and Sandford in [10] suggested the use of ToS field for covert channel creation. The last two bits of this field are reserved bits which can be used to carry secret data. Handel and Sandford also suggested the use of Timestamp option for covert communication. He suggested the use of a coding sequence to interpret Timestamp values i.e. even timestamp values should be interpreted as bit '0' and odd timestamps should be interpreted as bit '1'. So the last bit of the timestamp value is modified, if needed, to send the desired bit. This does not make much difference in timestamp value as well. In [16], Bharti et. al suggested the use of fields like padding bits, IP Identification, Source IP address having fake value to create covert channels over Internet Protocol. Zouheir and Imad in [19] proposed the use of record route option for covert communication over IPv4. Alsaffar & Johnson in [7] suggested the use of option length, pointer, overflow and flag bits of a Timestamp option to carry the covert data but as discussed above under IPv4 specifications, both length field and the pointer field can have a maximum value of 40 and if you try to assign a value to these fields which is greater than 40, a warning message is generated by Wireshark application (Wireshark is a freely available packet analyzer tool) [9] at the receiving end as shown in Figure 5. The warning messages are automatically highlighted by Wireshark in yellow. Moreover, if more covert messages like these are sent over the network, the generation of a large number of warning indications can draw the attention of a network administrator about some suspicious communication. In our work with timestamp option, we try to overcome this by using legitimate values for all the fields of Timestamp option and hiding data only in the Overflow field.

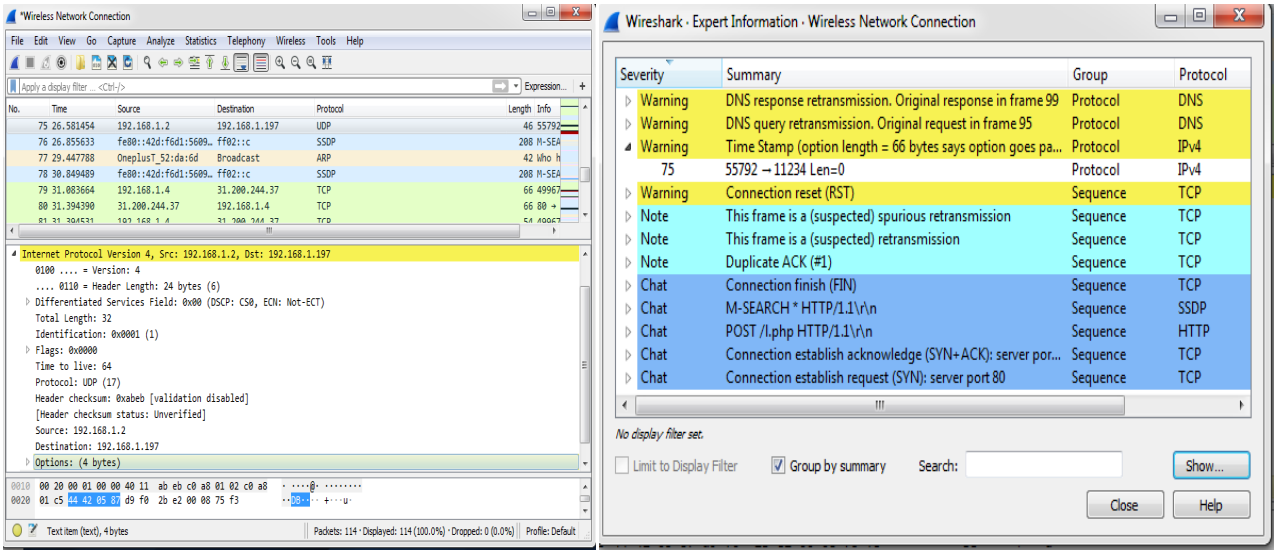

Fig. 5. Wireshark showing warning alerts.

4. Proposed System

In this paper, we propose a Network Steganography system that works over a LAN. The networking environment created for implementing Network Steganography is as shown in Figure 6. The LAN consisted of following devices:

- Router: The Router used in our system was HUAWEI – HG630V2 Home Gateway capable of both wired and wireless connections.
- Covert Message Sender (Host A): This is the device that wishes to covertly communicate with a receiver. This host makes use of Scapy library (version 2.4.3rc1) of Python to craft and send packets that carry covert data. The covert data is filled in the Overflow field of the Timestamp option of an IPv4 packet.
- Covert Message Receiver (Host B): This is the device that is waiting to receive secret data over the LAN from Host A (covert message sender). We chose UDP at the Transport layer for its well known advantage of being fast. If a user prefers, accuracy over speed, in that case UDP can be replaced by TCP at the transport layer. A UDP server is programmed and run over this system to listen for incoming UDP segment at a random port 11234. This entity after receiving a UDP segment at port 11234 scans the IP packet header inside this segment. It specifically looks and reads the Overflow field of Timestamp option of an IPv4 packet to fetch the covert message.
- Other Devices: The LAN was shared by other devices like laptops, computers, mobile phones etc.

All the devices in the LAN are either connected through wired or wireless connection to the Router. In our setup, as shown in Figure 6, Host A is the covert message sender and Host B is the covert message receiver. Host A takes 20 bits of input from the user, breaks these 20 bits into five groups of 4 bits each and crafts an IPv4 packet by adding five timestamp options with each timestamp option carrying four bits of data in the Overflow field. The Overflow field is normally used to carry the number of routers that were unable to add the timestamp value. In our technique, we have left enough space to add a maximum of five timestamp values. The local IP address of Host B is already known to Host A. It then sends this IP packet over UDP to Host B. We used UDP over TCP as the transport layer protocol as it is faster and does not require connection establishment.

On the receiver's side, a python script is executed which sniffs all the received packets and captures UDP segment received from Host A having destination port as 11234. From the captured packets, the script reads all the timestamp options of the IPv4 header and fetches the four bits of data from the Overflow field of each option. These four bits of data from each option are combined to read 20 bits of data sent by the covert Sender.

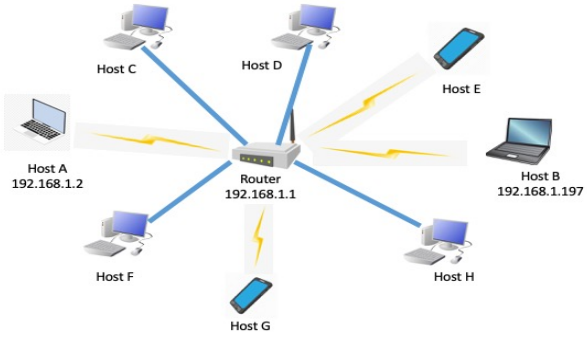

Fig. 6. Local Area Network Setup for Experiment.

5. Experimental Study

In this paper, we implement Network Steganography using the Overflow field of Timestamp option of an IPV4 packet. Scapy library is used to craft the IP packets. Scapy is a Python program that enables the user to send, sniff, dissect and forge network packets [5].
A communication setup is established between two devices A and B on a LAN connected to the Internet through Router R. A Python program is written to create a UDP segment over Internet Protocol version 4. The script on Host A crafts an IP packet using Scapy and adds covert information entered by the user, in the Overflow field of Timestamp option of an IPv4 packet. The Overflow field is 4 bit long and we can add a maximum of five Timestamp options with flag value set to 0 (maximum size of option field is ten 32-bit words) in a single IPv4 packet. So a single IPv4 packet can carry a maximum of 20 bits of covert data per packet.

We also experimented the idea of creating 10 timestamp headers with covert data in overflow field in each of the ten timestamp options, increasing the steganography bandwidth to 40 bits per packet. It worked well and covert data was correctly received at the receiver's side, but the drawback of this idea is that it did not leave any space for timestamps in the IPv4 header. Consequently, it may raise a suspicion about this IPv4 packet over the network. Moreover, if the router, through which this packet passes, tries to put its timestamp value in this header, it won't be able to do so as no space is left for that and further it will try to increment the value of the Overflow field (this field is incremented each time a router is unable to put its timestamp in an IPv4 header because of unavailability of space), so this may change our covert data. Hence, we propose to send five timestamp options with ample space for actual timestamp values, giving an overall steganography bandwidth of 20 bits/packet.

6. Results

The topology of the LAN setup for the experiment is shown in Figure 6. Here, Host A acts as a covert message sender and Host B acts as a covert message receiver. A UDP server script is created and executed at Host B to make it up and listening at a random port number 11234. Twenty bits of covert data per packet is transferred from Host A to Host B which can be verified from the snapshots of Wireshark in Figure 8. The snapshots of sending entity and receiving entity (UDP Server at port 11234) are also shown in Figure 7. The covert message is taken as input from the user, encapsulated in Timestamp option of IPv4 packet, and is sent to Host B whose IP address is already known over the LAN. This UDP packet is received correctly on the receiver side with rare packet loss. The Timestamp option of the IPv4 packet is then read by the receiver to fetch the covert data from the Overflow fields of five timestamp options correctly.

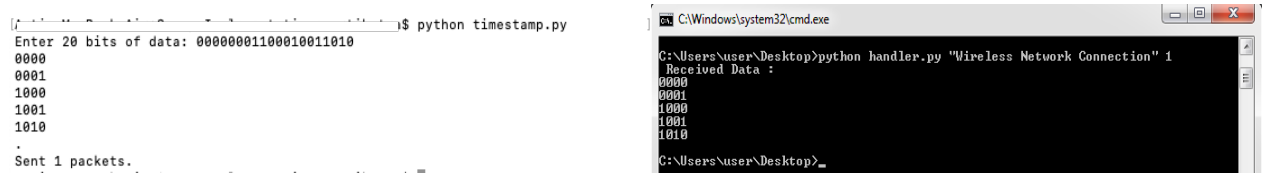

Fig. 7. Sender Process (Left) and Receiver Process (Right)

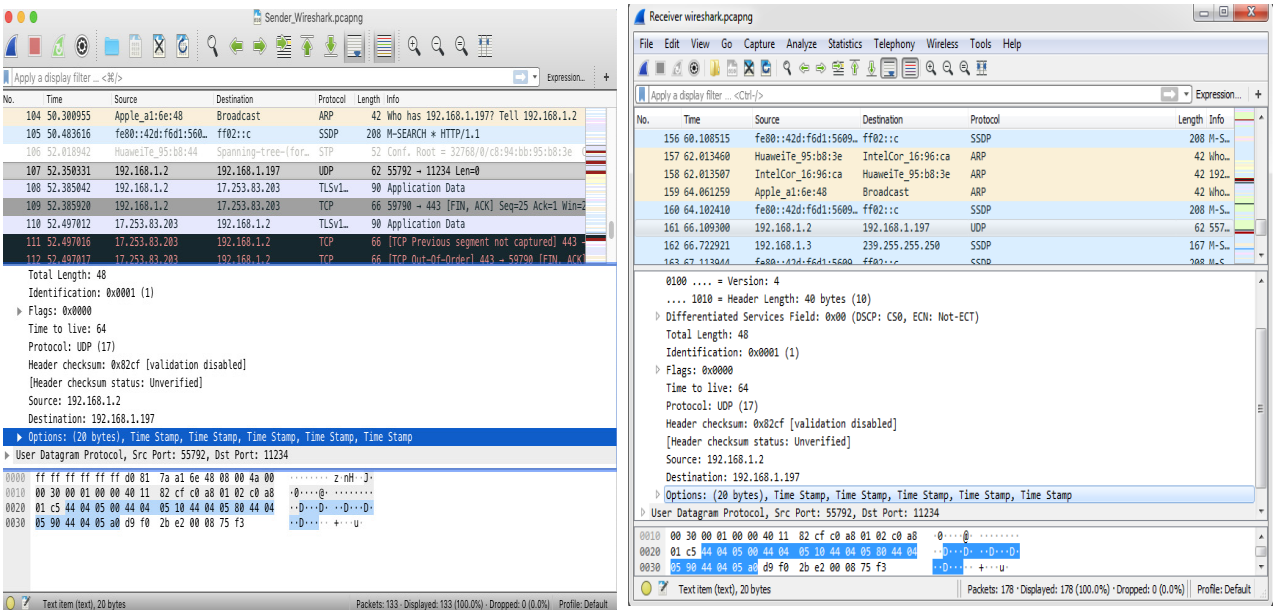

Fig. 8. Wireshark at Sender's (Left) and Receiver's (Right) machine.

7. Conclusion and Future Scope

In this paper, we proposed, developed and experimented a technique to implement Network Steganography using the Overflow field of timestamp option of an IPv4 packet. We created five timestamp options in a single IPv4 packet header. This channel was found to be capable of carrying 20 bits of data per packet. Our proposed scheme is better than the other network steganography scheme which uses timestamp option to carry covert data [7], in terms of undetectability as it sends 20 bits of data by using legitimate values for the Overflow field without generating any warning messages on Wireshark. The development and experimentation of this technique was solely done on a Local Area Network setup. The current system uses UDP at the transport layer, which is an unreliable delivery protocol and can be easily replaced by TCP if more reliability is needed. The advantage of our scheme is that it uses the very essential protocol which is the Internet Protocol for hiding the data. This provides us with a wide bandwidth of cover packets for sending covert data. Thus this work can further be extended to use the existing IP packets of the LAN instead of creating new ones to carry covert data. Future work can also be done to identify other option fields of IPv4 header to implement a covert channel and explore its feasibility over the Internet also.

References

[1] Szczypiorski, K. (2003). "Steganography in TCP/IP Networks. State of the Art and a Proposal of a New System-HICCUPS" *Warsaw University of Technology, Poland Institute of Telecommunications, Warsaw, Poland*.
[2] Fall, K. R., and W R. Stevens. (2011). "TCP/IP illustrated, volume 1: The protocols" *addison-Wesley*.
[3] ADVANCED INTERNET TECHNOLOGIES (IT – 302). IPv4 Header, https://advancedinternettechnologies.wordpress.com/ipv4-header/ (2012, accessed 02 Sept 2019).
[4] DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION. INTERNET PROTOCOL, https://tools.ietf.org/html/rfc791 (1981, accessed 02 Sept 2019).
[5] Introduction. About Scapy, https://scapy.readthedocs.io/en/latest/introduction.html, (2019, accessed 17 Sept 2019).
[6] Murdoch, Steven J. and Stephen Lewis. (2005) "Embedding Covert Channels into TCP/IP" in Barni M., Herrera-Joancomartí J., Katzenbeisser S., Pérez-González F. (eds) *Information Hiding, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg*.
[7] Alsaffar, Hassan and Daryl. (2015) "Covert channel using the IP timestamp option of an IPv4 packet." *The International Conference on Electrical and Bio-medical Engineering*: 48-51.
[8] A SPECIFICATION OF THE INTERNET PROTOCOL (IP) TIMESTAMP OPTION, https://tools.ietf.org/html/rfc781, (1981, accessed 18 Sept 2019).
[9] Wireshark-Go Deep, https://www.wireshark.org, (2019, accessed 19 Sept 2019).
[10] Handel, Theodore G, and Maxwell T. Sandford. (1996) "Hiding Data in the OSI Network Model" *Proceedings of 1st International Workshop, Information Hiding*: 23–38.
[11] IP option 4, Timestamp, http://www.networksorcery.com/enp/protocol/ip/option004.htm, (2018, accessed 23 Sept 2019).
[12] Ahsan, Kamran, and Deepa Kundur. (2002) "Practical data hiding in TCP/IP" *Proceedings of Workshop on Multimedia Security at ACM Multimedia*.
[13] Rowland, Craig H. (1997) "Covert channels in the TCP/IP protocol suite" *First Monday*, **2(5)**.
[14] Bellovin, Steven M. (1989) "Security problems in the TCP/IP protocol suite" *ACM SIGCOMM Computer Communication Review* **19 (2)**: 32-48.
[15] Kundur, Deepa and Kamran Ahsan. (2003) "Practical Internet Steganography: Data Hiding in IP" *Proceedings of Texas Workshop on Security of Information Systems, College Station, Texas.*
[16] Bharti, Vishal and Itu Snigdh. (2007) "Practical Development and Deployment of Covert Communication in IPv4" *Journal on Theoretical and Applied Information Technology*.
[17] Gupta, Richa, Sunny Gupta, and Anuradha Singhal. (2014) "Importance and techniques of information hiding: A review" *arXiv preprint* arXiv:1404.3063.
[18] Singh, Namrata, Jayati Bhardwaj, and Gunjan Raghav. (2017) "Network Steganography and its Techniques: A Survey" *International Journal of Computer Applications* **174 (2).**
[19] Trabelsi, Zouheir and Imad Jawhar. (2010) "Covert file transfer protocol based on the IP record route option." *Journal of Information Assurance and Security* **5 (1**): 64-73.
[20] Lubacz, Józef, Wojciech Mazurczyk, and Krzysztof Szczypiorski. (2014) "Principles and overview of network steganography." *IEEE Communications Magazine* **52 (5)**: 225-229.