INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 4, PP. 671-677
Manuscript received March 24, 2020; revised October, 2020. DOI: 10.24425/ijet.2020.134026

ARPNetSteg: Network Steganography Using Address Resolution Protocol

Punam Bedi, and Arti Dua

***Abstract*—Steganography is a technique that allows hidden transfer of data using some media such as Image, Audio, Video, Network Protocol or a Document, without its existence getting noticed. Over the past few years, a lot of research has been done in the field of Image, Video and Audio Steganography but very little work has been done in Network Steganography. A Network Steganography technique hides data in a Network Data Unit, i.e., a Network Protocol Packet. In this paper we present an algorithm ARPNetSteg that implements Network Steganography using the Address resolution protocol. Our technique is a robust technique that can transfer 44 bits of covert data per ARP reply packet.**

***Keywords*—Network Steganography, ARP Steganography, Covert Channel, Address Resolution Protocol, Protocol Steganography**

I. Introduction

INFORMATION hiding has been one of the favorite areas of researchers since the past two decades. Various techniques have been proposed by researchers to implement information security using information hiding. One such technique is steganography. Steganography hides data inside a cover medium without being perceived. The common media that are used to implement steganography are images, audios, videos, network protocols, documents etc. Steganography is implemented by modulating those characteristics of the media that do not bring perceptible changes in the original media. The changes in characteristics are made as per the secret message. For example, in images the Least Significant Bit (LSB) of some or all pixels is modulated as per the secret message. If the secret bit matches with the LSB of the current pixel then no change is made otherwise the LSB is toggled. Many techniques have been proposed in literature for Image, Audio and Video Steganography, whereas very less work has been done in the field of Network Steganography. In this paper, we propose a Network Steganography technique which hides data in a Network data unit that is a Network Protocol Packet. A Network Protocol Packet is a logical entity that carries data over a network. This Network packet has two major components: a) Protocol Header b) Payload [1]. The protocol header carries all the control information relevant to this packet whereas the payload carries the actual data to be transferred over the network. Some of the vital protocols of networks are Internet Protocol, Address Resolution Protocol, Transmission Control Protocol etc. In our proposed technique, we make use of an Address Resolution Protocol (ARP) Packet to carry secret message over the network. ARP is responsible for mapping network layer address to the hardware address of a device. Since ARP operates in a Local Area Network (LAN) setup, our technique is implemented and experimented solely over a LAN.

Rest of the paper is organized as follows. Section II describes the background, working and format of Address Resolution Protocol. This section also briefly describes the concept of ARP spoofing, which is a well known vulnerability of networks. Section III gives brief overview of related work done in the field of Network Steganography using the Address Resolution Protocol. Section IV elaborates the proposed system. Section V describes the experimental study and results of experiments conducted on ARPNetSteg and section VI concludes the paper.

II. Background

The word Steganography was defined by Trithemius in his book Steganographia [2]. The term Steganography was derived from two greek words, *steganos* that means hidden and *graphia* that means writing. Steganography is defined as undetectably altering a work to embed a secret message [3]. Steganography aims at hiding the very existence of a secret message. To conceal a message, a cover media is required. This cover media can be a document, an image, an audio, a video or a network protocol. Steganography is achieved by hiding the secret message into a cover message as shown in Fig. 1.

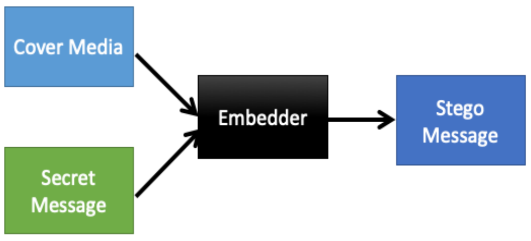


Fig. 1. Steganography Process

Network Steganography or Protocol Steganography, one of the classifications of Steganography has recently captured lot of the attention from researchers. The term Network steganography was used for the very first time by K. Szczypiorski in 2003 [4]. In this type of steganography, one tries to hide secret data in parts of the network packets carrying normal communication data over the networks without getting noticed. Due to increased use of internet and growth in high speed network technologies, even if one bit of information can be hidden and transferred in a packet, a large and popular website could lose approximately 26 GB of data annually [5].

A network packet broadly consists of a header and payload. Header carries the control information of the packet whereas the Payload carries the actual information that needs to be

Punam Bedi and Arti Dua are with University of Delhi, Delhi, India. (e-mail: punambedi@ieee.org, arti.batra@gmail.com)

transferred. Network Steganography exploits both header and payload individually or together to hide the secret data. Network Steganography techniques can be classified into three categories:

(1) Storage Based Techniques: These techniques hide data in the storage part of a PDU, which is either a packet's header, or payload or both.

(2) Timing Based Techniques: These techniques use the sequence numbers, delays or inter-packet timing interpretations to send secret data.

(3) Hybrid Techniques: These techniques use a combination of both Storage based techniques and Timing based techniques to send the covert data.

In this paper, we exploit the Address Resolution Protocol [6] using a Storage based channel for sending covert data.

A. *Address Resolution Protocol*

The Address Resolution Protocol is a vital protocol of a Local Area Network (LAN) that is responsible for mapping a network protocol address to 48 bits Ethernet address for transmission on Ethernet hardware. The complete description of ARP was given by David C. Plummer in RFC 826 [6] in November 1982. The format of an ARP packet header is as shown in Fig. 2.

Hardware Type (2 Bytes)		Protocol Type (2 Bytes)
Hardware Address Length (1 Byte)	Protocol Address Length (1 Byte)	Operation (2 Bytes)
Sender Hardware Address (4 Bytes)		
Sender Hardware Address (2 Bytes)		Sender IP Address (2 Bytes)
Sender IP Address (2 Bytes)		Target Hardware Address (2 Bytes)
Target Hardware Address (4 Bytes)		
Target IP Address (4 Bytes)		

Fig. 2. ARP Header Format

The Hardware Type defines the protocol being used at data link layer. For Ethernet, the value of this field is 1. For IEEE 802., the value of this field is 6. The value 0 is reserved [6]. The Protocol Type field specifies the internetwork protocol for which ARP request is generated. For example, for Internet Protocol Version 4 (IPv4), the value of Protocol Type is 0x0800. Hardware address length field specifies the number of octets in hardware address. For example, for Ethernet address, the hardware address length is 6. Protocol Address length specifies the number of octets in Protocol Address. For example, for IPv4 the protocol address length is 4. Operation field specifies the action being performed. Its value is set to 1 for ARP Request procedures while it is set to 2 for ARP Reply procedures. The Sender Hardware Address field holds the hardware address of the source. This field is variable in length and equals to the value in Hardware Address Length field. It is 6 bytes in case of Ethernet as shown in Fig. 2. The Sender IP address holds the IP address of the source. This field is also variable in length and equals the value in Protocol Length Field [7, 8]. The Target Hardware Address field holds the hardware address of the destination. This field is again variable in length and equals the value in the Hardware Address Length field. The Target IP address holds the IP address of the destination. Just like Sender IP Address field, this field is also variable in length and equals the value in Protocol Length Field.

B. *How ARP works?*

The Internet protocol provides interoperability of packet switching across a large variety of physical network types [8]. It requires a mapping between the physical address at link layer and logical address at the network layer. ARP provides this mapping among various address types. When two hosts want to communicate over a network, they should know both each other's hardware address and the IP address. ARP is a vital network protocol in a local area network that helps a node to identify the hardware address of another node whose IP address is already known. For this, the host that needs to identify the hardware address of other device, sends an ARP Broadcast Request over the network as shown in Fig 3. This Broadcast Request contains the IP address of the node whose machine address is needed. The response to this ARP request is generated by a machine whose IP address matches the one in the Target IP Address field of the ARP request packet. The machine writes its hardware address in this ARP reply packet and unicasts it to the original source as shown in Fig 4.

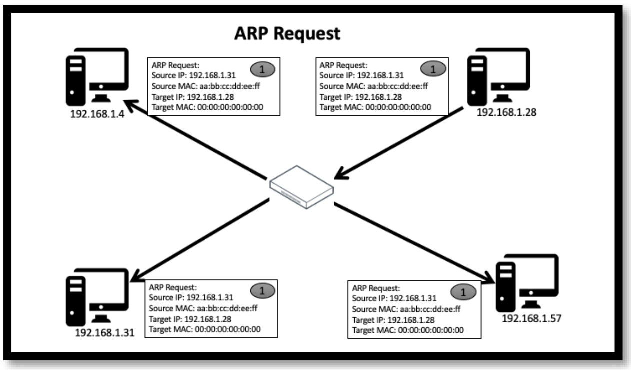

ARP Request
192.168.1.4
ARP Request: Source IP: 192.168.1.31 Source MAC: aa:bb:cc:dd:ee:ff Target IP: 192.168.1.28 Target MAC: 00:00:00:00:00:00
1
ARP Request: Source IP: 192.168.1.31 Source MAC: aa:bb:cc:dd:ee:ff Target IP: 192.168.1.28 Target MAC: 00:00:00:00:00:00
1
192.168.1.28
192.168.1.31
ARP Request: Source IP: 192.168.1.31 Source MAC: aa:bb:cc:dd:ee:ff Target IP: 192.168.1.28 Target MAC: 00:00:00:00:00:00
1
ARP Request: Source IP: 192.168.1.31 Source MAC: aa:bb:cc:dd:ee:ff Target IP: 192.168.1.28 Target MAC: 00:00:00:00:00:00
1
192.168.1.57

Fig. 3. ARP Request Process

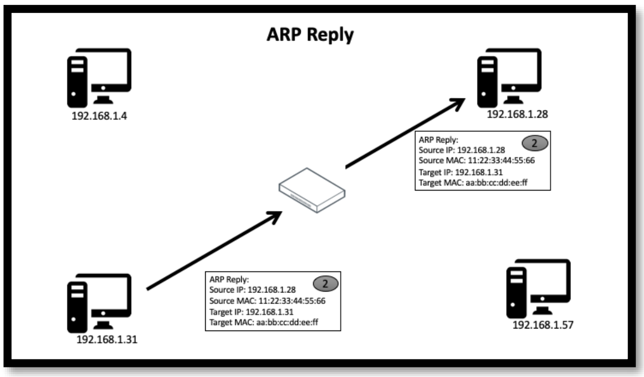

ARP Reply
192.168.1.4
192.168.1.28
ARP Reply: Source IP: 192.168.1.28 Source MAC: 11:22:33:44:55:66 Target IP: 192.168.1.31 Target MAC: aa:bb:cc:dd:ee:ff
2
ARP Reply: Source IP: 192.168.1.28 Source MAC: 11:22:33:44:55:66 Target IP: 192.168.1.31 Target MAC: aa:bb:cc:dd:ee:ff
2
192.168.1.31
192.168.1.57

Fig. 4. ARP Reply Process

C. *ARP Spoofing*

The Address Resolution Protocol provides no mechanism for authenticating the source address of a machine sending an ARP response. A proxy ARP is a system which replies on behalf of some other system, normally as a part of network design [8]. On the other hand, in ARP spoofing, a spoofer responds to an ARP request sent for another system, mostly with a malicious intent to intercept the data being sent to that authentic system. ARP is a stateless protocol, i.e. all the nodes over a network maintain an individual copy of ARP cache table stored in their own devices. This table stores the paired entries of IP addresses and their corresponding MAC addresses for the nodes in the network. This table is created and updated as and when a new ARP reply is received, regardless of whether any ARP request is made by this node or not. Whenever a new ARP reply is received for an existing entry, this entry is updated even if the older entry has yet not expired. In our proposed method, we partially make use of this technique, by spoofing for local IP addresses which are not presently being used by any node over a LAN.

III. Related Work

Since past few years, researchers have been working in the field of Network Steganography. Many Network Steganography techniques have been proposed and developed in protocols used in TCP/IP Networks. Handel and Stanford [9] discussed various covert channels possible in OSI model. Authors in [10-13] proposed various techniques to develop covert channels using the protocols used in TCP/IP model. K. Szczypiorski, M. Drzymała, and M. Ł. Urbański [14] proposed a covert channel using the DNS protocol which works at the application layer. Z. Trabelsi and I. Jawhar [15] exploited the options field of IP header to develop a covert channel using the record route option. P.Bedi and A. Dua [16] used another option field called timestamp to implement a covert channel using IPv4 protocol. As per our knowledge, very few researchers have exploited the Address Resolution Protocol for Steganography. L. Ji, Y. Fan and C. Ma [17] proposed a covert channel using ARP. In their technique, they used target IP address field to encode the covert information. They used the last ‘t’ bits (where t lies between 4 and 8) to store the secret information. To identify the ARP requests carrying covert information, the first ‘8-t’ bits of the last byte that encodes the sending second, are first inverted and then XORed with first ‘8-t’ covert bits of the first ‘t’ covert bits. B. Jankowski, W. Mazurczyk and K. Szczypiorski [18] proposed a method called PadSteg. This method implements inter-protocol network steganography (a technique that uses more than one protocol for implementing network steganography) which uses ARP and other protocols like TCP or ICMP to exploit Etherleak vulnerability to provide secret communications between a group of nodes in a LAN. Schmidbauer, Tobias, Steffen Wendzel, Aleksandra Mileva, and Wojciech Mazurczyk [19] used the concepts of dead drops in ARP caches maintained at a host and SNMP to store and read the covert data respectively. A third party node in a LAN is used as a dead drop. The Covert message sender drops covert information in this node’s ARP Cache and a Covert Message Receiver retrieves covert information from this ARP Cache using SNMP protocol. In our technique, we use intra-protocol steganography (Protocol steganography technique that uses a single network protocol) using ARP protocol that partially uses the concept of ARP spoofing to transfer covert data over a local area network. Our scheme provides a bandwidth of 44 bits per ARP reply packet.

IV. Proposed Technique

In this paper, we propose a technique that uses Address Resolution Protocol to implement Network Steganography. This technique works on a Local Area Network that has one covert message sender (Host A) and one covert message receiver (Host B). A series of steps at Host A and Host B are executed simultaneously to implement this technique.

A. *Sender Side Algorithm*

To begin with, we input a covert message string and compute its length. Next, we encode the covert message by converting the message string to hexadecimal code. After that we scan the local area network for free or unallocated local IP addresses (local IP addresses that are not being used over this LAN). This list is created by sending a broadcast request for all possible local IP addresses and then waiting for their respective ARP replies. The local IP addresses for which we receive ARP replies are assumed to be allocated ones and the rest for which no ARP replies are received, are added to the unallocated list. This step may be repeated to make sure that the list does not mistakenly add an allocated IP address whose reply is not received because of getting lost over the network. In the next step, we enter a seed value for random local IP address generation. The purpose of a seed is to generate the same set of random numbers for a given continuous set of numbers. In our case, we use same pre-known seed value at the sender and receiver side to generate same set of random local IP addresses from the unallocated list at both ends. A random number ‘x’, between 1 and 255 is generated with the common seed value described in the previous step. Further, we check if the address 192.168.1.x is present in the unallocated list or not. In case this local IP address is not present in the unallocated list, the next random number is generated with the same seed value. This is repeated till the time we get an unallocated local IP address. Once an unallocated local IP address is found, the covert message sender waits for an ARP broadcast request from the covert message receiver (Host B) for this local IP address.

After the awaited ARP request is received from the covert message receiver (Host B), the covert message sender begins to create an ARP reply for the same. It puts the first eleven or lesser (if message is smaller) hexadecimal digits of the covert message in the Sender Hardware Address field of ARP reply message. If covert message length is less that eleven hex-digits, these hex-digits are padded to make it eleven hex-digits long. The last hexadecimal digit (quad) of the Ethernet address is used to store the control information (control quad). This control quad mainly carries the information if the message being carried is over or more message data needs to be sent in later packets. That is, this control quad decides whether more ARP requests are to be generated by the covert message receiver or not. If more message data is left to send at sender’s site, it sets the value of control quad as 0x0. Otherwise, if no more message data is left, the number of hex-digits filled in ARP reply field excluding padding is calculated. If its value is exactly 11, control quad is set to 0xf else control quad is set as the number of hex digits used for padding in first eleven digits of Sender Hardware Address field. This sequence of steps from generation of random numbers to successful sending of ARP reply is repeated till complete covert message is sent from the sender side. The sender side algorithm is shown in Fig 5.

Input: A Covert message string S, a seed value SV.

```
1.  Input a covert message string S from user
2.  Convert the message string to hexadecimal code $S_x$
3.  For each possible local IP address on this LAN
4.      Create an ARP Request
5.      Broadcast it over the local area network
6.  End
7.  For each ARP reply received
8.      Add the Source IP address of this reply to allocated
        list AL.
9.  End
10. For all possible local IP address for this LAN absent in AL
11.     Add the local IP addresses to unallocated list UL
12. End
13. Use seed SV to generate next random number x,
    where $x \in (1,255)$
14. Create a Local IP address A as 192.168.1.x
15. If address A does not belong to UL,
16.     Goto step 13.
17. Wait for an ARP Request
18. If Req. Received==ARP Req. && ARP{Target IP}==
    192.168.1.x && ARP{Source IP}==IP of Covert
    Receiver, then
19.     Create an ARP reply with Source IP =
        192.168.1.x, Target IP = Source IP value received
        in ARP request and Target Hardware Address =
        Sender Hardware address value received in ARP
        request
20.     Pick first eleven or lesser (if less digits are left) Hex-
        digits from $S_x$, call it $S_{11}$
21.     Update $S_x$ as $S_x - S_{11}$ .
22.     Add $S_{11}$ in the first eleven Hex digits of Sender
        Hardware Address field of ARP reply
23.     If $S_x$ is not empty
24.         Set the twelfth hex digit of Sender Hardware-
            Address field (control quad) in ARP reply as 0x0
25.     Else
26.         If length($S_{11}$) < 11
27.             Compute 11 – length($S_{11}$), call it pad_num
28.             Pad the last pad_num number of hex digits
                in Sender Hardware Address field with
                value 0x0 for each hex digit.
29.             Set the twelfth digit (control quad) of Source
                Hardware Address field of ARP reply as
                pad_num in hexadecimal.
30.         Else
31.             If length($S_{11}$) == 11
32.                 Set the twelfth digit (control quad)
33.                 of Sender Hardware Address field
                    of ARP reply as 0xf.
34. If $S_x$ is empty
35.     Exit
36. Else
37.     Go to Step 13
```

Fig. 5. Sender Side Algorithm

B. *Receiver Side Algorithm*

Firstly, scan the LAN for unallocated local IP addresses. The list of unallocated local IP addresses is created exactly the same way as done in the sender side algorithm. Next, value of seed for random local IP address generation is inputted. Both receiver and sender enter the same seed value known to them in prior. After that a random number 'x' between 1 and 255 is generated with the same seed value described in the previous step. Further, we check if the IP address 192.168.1.x is present in the unallocated list or not. In case, this local IP address is not present in the unallocated list, the next random number is generated with the same seed value. This is repeated till we get an unallocated local IP address. Once an unallocated local IP address is found, the covert message receiver creates an ARP Broadcast Request with this randomly generated local IP address as value for Target IP Address field.

Input: A seed value SV

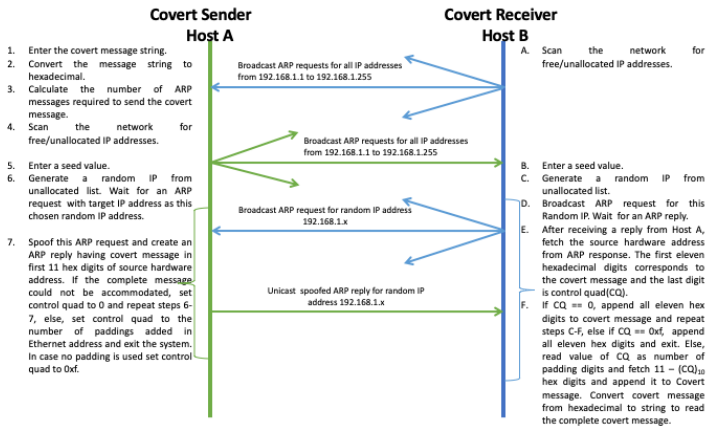
```
1.  Initialize Covert_MSG string to NULL
2.  For each possible local IP address on this LAN
3.      Create an ARP Request
4.      Broadcast it over the local area network
5.  End
6.  For each ARP reply received
7.      Add the source IP address to allocated list AL.
8.  End
9.  For all possible local IP addresses for this LAN, not
    present in AL
10.     Add the local IP addresses to unallocated list UL
11. End
12. Use seed SV to generate next random number x,
    where $x \in (1,255)$
13. Create a Local IP address A as 192.168.1.x
14. If address A does not belong to UL,
15.     Go to step 12
16. Create an ARP Request with Target IP address as
    192.168.1.x
17. Wait for an ARP reply
18. If Req. Received==ARP Reply and ARP{Source IP}==
    192.168.1.x, then
19.     Fetch the Sender Hardware Address field as SHF
20.     Set Control Quad, CQ = twelfth hex digit of SHF
21.     If CQ == 0x0
22.         Pick first eleven Hex-digits from SHF and
            append it to Covert_MSG string
23.         Goto step 12
24.     Else
25.         If CQ == 0xf
26.             Pick first eleven Hex-digits from SHF and
                append it to Covert_MSG string.
27.         Else
28.             Convert CQ to decimal and call it $CQ_{10}$
29.             Set temp = 11 – $CQ_{10}$
30.             Pick first temp digits from SHF and append
                it to Covert_msg string
31. Convert Covert_MSG (in hex) to string and read it as
    complete covert string
32. Exit
```

Fig. 6. Receiver Side Algorithm

After this ARP request is broadcasted, Host B enters a wait state waiting for a spoofed ARP reply for this request from covert message sender (Host A) only, as no node exists on this network with the same local IP Address as per the unallocated list created in previous step. In this way covert message sender partially abuses ARP spoofing and creates and sends an ARP reply. Once an ARP Reply is received at this host for the ARP Request just sent, the receiver checks for the Sender Hardware Address field in the received ARP response. The first eleven hex digits carry the covert data with/without padding.

The twelfth hexadecimal digit or the last quad in this Ethernet address is the control quad which tells two things. Firstly, if the sender wishes to send more data or this is the last message and secondly, if there is any padding in the first eleven hex digits or not. If the value of this control bit is 0x0, it means that the sender has more data to send and no padding is used in eleven hex digits. Following that, steps beginning from generation of random IP till processing of the received ARP reply, are repeated till the value of control quad received in the ARP reply is non zero. If the value of control quad is 0xf in hexadecimal, it is interpreted as the last data message with zero padding (meaning all eleven quads hold relevant covert message data). And if the value of control bit is anything other than 0 and 0xf, it is interpreted as last data message with the value of control quad signifying the number of quads that contain padded data. For example, if the value of control quad is 0xa, it means out of the eleven data quads, last ten quads hold padded value and only the first quad has relevant covert data value. The receiver side algorithm for this technique is given in Fig 6.

Figure 7 shows the time and flow diagram of our proposed technique.


Covert Sender
Host A
Covert Receiver
Host B
Broadcast ARP requests for all IP addresses from 192.168.1.1 to 192.168.1.255
Broadcast ARP requests for all IP addresses from 192.168.1.1 to 192.168.1.255
Broadcast ARP request for random IP address 192.168.1.x
Unicast spoofed ARP reply for random IP address 192.168.1.x

1. Enter the covert message string.
2. Convert the message string to hexadecimal.
3. Calculate the number of ARP messages required to send the covert message.
4. Scan the network for free/unallocated IP addresses.
5. Enter a seed value.
6. Generate a random IP from unallocated list. Wait for an ARP request with target IP address as this chosen random IP address.
7. Spoof this ARP request and create an ARP reply having covert message in first 11 hex digits of source hardware address. If the complete message could not be accommodated, set control quad to 0 and repeat steps 6-7, else, set control quad to the number of paddings added in Ethernet address and exit the system. In case no padding is used set control quad to 0xf.

A. Scan the network for free/unallocated IP addresses.
B. Enter a seed value.
C. Generate a random IP from unallocated list.
D. Broadcast ARP request for this Random IP. Wait for an ARP reply.
E. After receiving a reply from Host A, fetch the source hardware address from ARP response. The first eleven hexadecimal digits corresponds to the covert message and the last digit is control quad(CQ).
F. If CQ == 0, append all eleven hex digits to covert message and repeat steps C-F, else if CQ == 0xf, append all eleven hex digits and exit. Else, read value of CQ as number of padding digits and fetch 11 – $(CQ)_{10}$ hex digits and append it to Covert message. Convert covert message from hexadecimal to string to read the complete covert message.

Fig. 7. Flow Diagram of ARPNetSteg.

V. EXPERIMENTAL STUDY

A. Implementation

To implement and test ARPNetSteg, we created a Local Area Network which consisted of a covert message sender (Host A) and a covert message receiver (Host B) who wish to communicate over a LAN. The local IP address of Host A was 192.168.1.4. The local IP address of Host B was 192.168.1.14. There were many other devices connected to Router. The local IP address of the Router was 192.168.1.1 in this LAN.

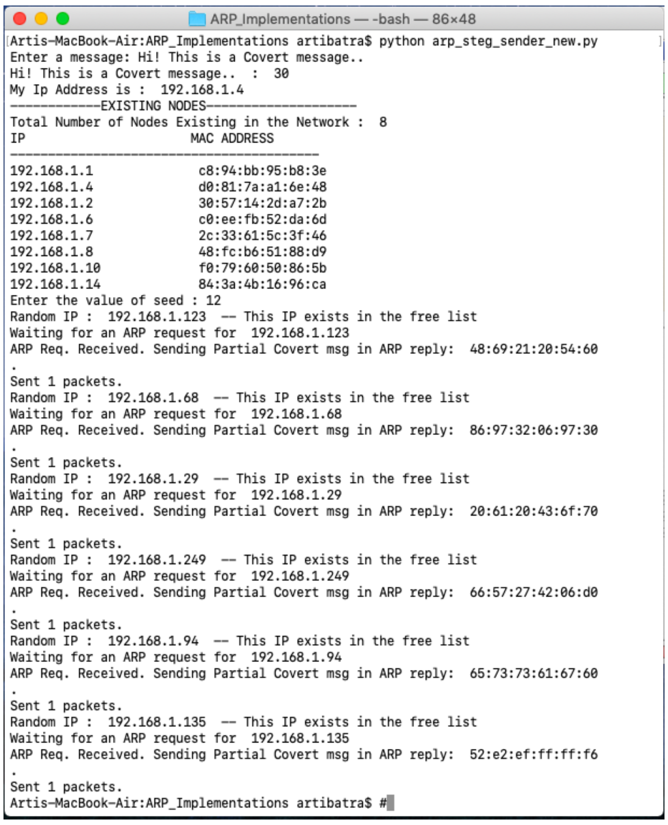
In the first step, both Host A and Host B created a list of unused or unallocated local IP addresses in this LAN. After the successful creation of this list, a common seed value (known in prior to both the sender and the receiver) was entered. The purpose of this seed was to pick same order of random local IP addresses from the unallocated list (created in the previous step) for communication between the sender and the receiver. After choosing a random local IP address, the covert message receiver (Host B) created and sent an ARP broadcast request having Target IP Address set to this randomly selected local IP address value from the previous step. Scapy [20] library with Python was used to create and send ARP packets. On the other side, the covert message sender (Host A) after generating its list of unallocated local IP addresses, waited for a Broadcast Request for the same target IP address (generated in previous step with the same seed value). As the Broadcast Request for this IP address was received by the covert message sender, it created a spoofed ARP reply for this request. Since there was no node over the LAN which had this same local IP address, hence only covert message sender responded to this ARP Broadcast Request. In this ARP reply message, the covert message sender entered the covert data in the Source Hardware Address field. In case of Ethernet, the hardware address is 6 Bytes or 48 bits long. The covert message sender stored the message in the first 44 bits of Ethernet address, encoded in hexadecimal notation. The last quad (4 bits) as intended was used as control quad. The length of covert message is used to decide the number of ARP requests required to transfer the complete message. If the covert message could be accommodated in less than first 44 bits, then the value of control quad was set to the number of padding hexadecimal digits added to the message to make it 44 bits long. Otherwise, if covert message used all the 44 bits then the value of control quad was set to 0xf in hexadecimal. If more than 44 bits was required to send the complete message, the control quad was set to 0x0. After the successful creation of this ARP reply packet a unicast ARP reply was sent to Host B, who is the covert message receiver. This receiver on receiving this message extracted data from Sender Hardware Address field of the ARP reply header and further checked the control quad. If the value of control quad was 0x0, the first eleven hex-digits were stored as covert data and also it was interpreted that the covert message sender wants to send more data, consequently it sent another Broadcast ARP Request using Target IP address field value as the second unallocated random IP value generated with the same seed. However, if the value of control quad was non-zero, there were two interpretations. If its value was 0xf, it interpreted it as the last message data with zero padding. And if the value of control quad was anything other than 0 and 0xf, it was interpreted as last message with the value of control quad signifying the number of quads that contain padded data. Depending on the value of control quad the covert message was fetched accordingly. Further, to capture the sent and received packets at both sender and receiver, Wireshark [21] packet analyzer was used. Wireshark is a freely available tool that is used for analysis of packets flowing over any network.

B. Results

The technique demonstrated in Section IV was implemented and experimented over a Local Area Network. We entered a secret message at Host A and it was successfully transferred to Host B using ARP Request and ARP Reply messages. Figure 8 and 10 show the console snapshots of successful sending and receiving of desired ARP messages at sending and receiving devices respectively.

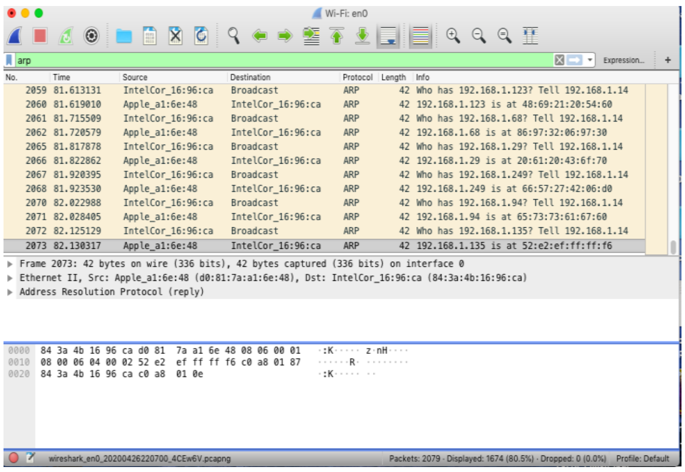
The covert message entered at the sender site was “Hi! This is a Covert message..”, which was 30 characters long as shown in Fig. 8. Hence it required six ARP requests and reply pairs to communicate covertly with our technique. Wireshark tool was used to capture packets coming in and going out of Host A and Host B. A screenshot of ARP packets carrying covert data from Host A in six ARP replies corresponding to ARP requests received from Host B is shown in Fig 9 using Wireshark.

Figure 11 shows another Wireshark screenshot that captured ARP request packets sent from receiver side (Host B) and the corresponding ARP replies received with covert data at the receiver side (Host A). More Experiments were conducted to send different covert messages and all reached the destination accurately.



```
Artis-MacBook-Air:ARP_Implementations artibatra$ python arp_steg_sender_new.py
Enter a message: Hi! This is a Covert message..
Hi! This is a Covert message..  :  30
My Ip Address is :  192.168.1.4
------------EXISTING NODES--------------------
Total Number of Nodes Existing in the Network :  8
IP                       MAC ADDRESS
---------------------------------------------
192.168.1.1              c8:94:bb:95:b8:3e
192.168.1.4              d0:81:7a:a1:6e:48
192.168.1.2              30:57:14:2d:a7:2b
192.168.1.6              c0:ee:fb:52:da:6d
192.168.1.7              2c:33:61:5c:3f:46
192.168.1.8              48:fc:b6:51:88:d9
192.168.1.10             f0:79:60:50:86:5b
192.168.1.14             84:3a:4b:16:96:ca
Enter the value of seed : 12
Random IP :  192.168.1.123  -- This IP exists in the free list
Waiting for an ARP request for  192.168.1.123
ARP Req. Received. Sending Partial Covert msg in ARP reply:  48:69:21:20:54:60
.
Sent 1 packets.
Random IP :  192.168.1.68  -- This IP exists in the free list
Waiting for an ARP request for  192.168.1.68
ARP Req. Received. Sending Partial Covert msg in ARP reply:  86:97:32:06:97:30
.
Sent 1 packets.
Random IP :  192.168.1.29  -- This IP exists in the free list
Waiting for an ARP request for  192.168.1.29
ARP Req. Received. Sending Partial Covert msg in ARP reply:  20:61:20:43:6f:70
.
Sent 1 packets.
Random IP :  192.168.1.249  -- This IP exists in the free list
Waiting for an ARP request for  192.168.1.249
ARP Req. Received. Sending Partial Covert msg in ARP reply:  66:57:27:42:06:d0
.
Sent 1 packets.
Random IP :  192.168.1.94  -- This IP exists in the free list
Waiting for an ARP request for  192.168.1.94
ARP Req. Received. Sending Partial Covert msg in ARP reply:  65:73:73:61:67:60
.
Sent 1 packets.
Random IP :  192.168.1.135  -- This IP exists in the free list
Waiting for an ARP request for  192.168.1.135
ARP Req. Received. Sending Partial Covert msg in ARP reply:  52:e2:ef:ff:ff:f6
.
Sent 1 packets.
Artis-MacBook-Air:ARP_Implementations artibatra$ #
```

Fig. 8. Console at Host A (Covert Message Sender).



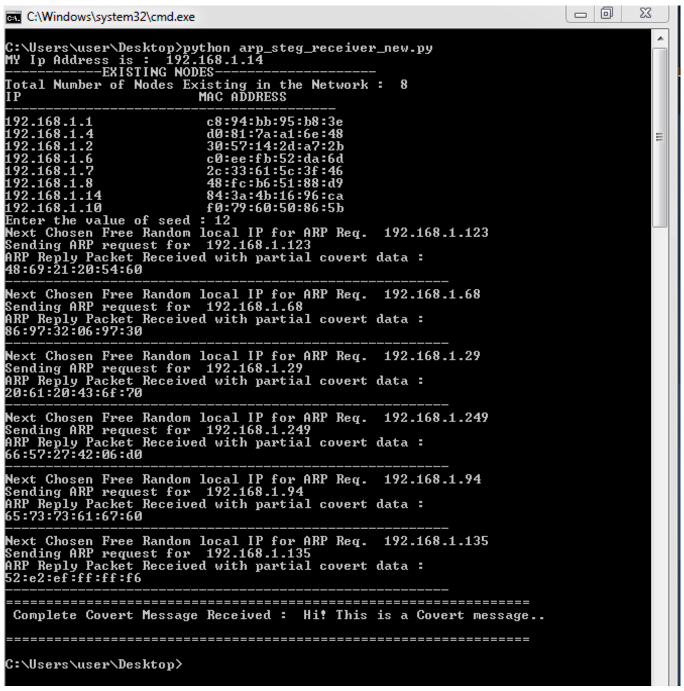
```
No.    Time       Source              Destination         Protocol Length Info
  2059 81.613131  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.123? Tell 192.168.1.14
  2060 81.619010  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.123 is at 48:69:21:20:54:60
  2061 81.715509  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.68? Tell 192.168.1.14
  2062 81.720579  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.68 is at 86:97:32:06:97:30
  2065 81.817878  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.29? Tell 192.168.1.14
  2066 81.822862  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.29 is at 20:61:20:43:6f:70
  2067 81.920395  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.249? Tell 192.168.1.14
  2068 81.923530  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.249 is at 66:57:27:42:06:d0
  2070 82.022988  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.94? Tell 192.168.1.14
  2071 82.028405  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.94 is at 65:73:73:61:67:60
  2072 82.125129  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.135? Tell 192.168.1.14
  2073 82.130317  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.135 is at 52:e2:ef:ff:ff:f6

▶ Frame 2073: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▶ Ethernet II, Src: Apple_a1:6e:48 (d0:81:7a:a1:6e:48), Dst: IntelCor_16:96:ca (84:3a:4b:16:96:ca)
▶ Address Resolution Protocol (reply)

0000  84 3a 4b 16 96 ca d0 81  7a a1 6e 48 08 06 00 01   ·:K····· z·nH····
0010  08 00 06 04 00 02 52 e2  ef ff ff f6 c0 a8 01 87   ······R· ········
0020  84 3a 4b 16 96 ca c0 a8  01 0e                     ·:K····· ··

wireshark_en0_20200426220700_4CEw6V.pcapng      Packets: 2079 · Displayed: 1674 (80.5%) · Dropped: 0 (0.0%)   Profile: Default
```

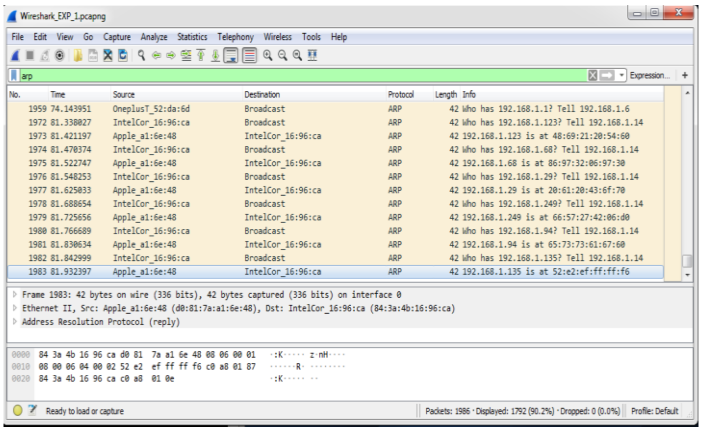
Fig. 9. Wireshark snapshot at Sender Site



```
C:\Users\user\Desktop>python arp_steg_receiver_new.py
MY Ip Address is :  192.168.1.14
------------EXISTING NODES--------------------
Total Number of Nodes Existing in the Network :  8
IP                         MAC ADDRESS
---------------------------------------------
192.168.1.1                c8:94:bb:95:b8:3e
192.168.1.4                d0:81:7a:a1:6e:48
192.168.1.2                30:57:14:2d:a7:2b
192.168.1.6                c0:ee:fb:52:da:6d
192.168.1.7                2c:33:61:5c:3f:46
192.168.1.8                48:fc:b6:51:88:d9
192.168.1.14               84:3a:4b:16:96:ca
192.168.1.10               f0:79:60:50:86:5b
Enter the value of seed : 12
Next Chosen Free Random local IP for ARP Req.  192.168.1.123
Sending ARP request for  192.168.1.123
ARP Reply Packet Received with partial covert data :
48:69:21:20:54:60
---------------------------------------------
Next Chosen Free Random local IP for ARP Req.  192.168.1.68
Sending ARP request for  192.168.1.68
ARP Reply Packet Received with partial covert data :
86:97:32:06:97:30
---------------------------------------------
Next Chosen Free Random local IP for ARP Req.  192.168.1.29
Sending ARP request for  192.168.1.29
ARP Reply Packet Received with partial covert data :
20:61:20:43:6f:70
---------------------------------------------
Next Chosen Free Random local IP for ARP Req.  192.168.1.249
Sending ARP request for  192.168.1.249
ARP Reply Packet Received with partial covert data :
66:57:27:42:06:d0
---------------------------------------------
Next Chosen Free Random local IP for ARP Req.  192.168.1.94
Sending ARP request for  192.168.1.94
ARP Reply Packet Received with partial covert data :
65:73:73:61:67:60
---------------------------------------------
Next Chosen Free Random local IP for ARP Req.  192.168.1.135
Sending ARP request for  192.168.1.135
ARP Reply Packet Received with partial covert data :
52:e2:ef:ff:ff:f6
---------------------------------------------
=============================================================
 Complete Covert Message Received :  Hi! This is a Covert message..
=============================================================

C:\Users\user\Desktop>
```

Fig. 10. Console at Host B (Covert Message Receiver)



```
No.    Time       Source              Destination         Protocol Length Info
  1959 74.143951  OneplusT_52:da:6d   Broadcast           ARP      42 Who has 192.168.1.1? Tell 192.168.1.6
  1972 81.338027  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.123? Tell 192.168.1.14
  1973 81.421197  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.123 is at 48:69:21:20:54:60
  1974 81.470374  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.68? Tell 192.168.1.14
  1975 81.522747  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.68 is at 86:97:32:06:97:30
  1976 81.548253  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.29? Tell 192.168.1.14
  1977 81.625033  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.29 is at 20:61:20:43:6f:70
  1978 81.688654  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.249? Tell 192.168.1.14
  1979 81.725656  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.249 is at 66:57:27:42:06:d0
  1980 81.766689  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.94? Tell 192.168.1.14
  1981 81.830634  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.94 is at 65:73:73:61:67:60
  1982 81.842999  IntelCor_16:96:ca   Broadcast           ARP      42 Who has 192.168.1.135? Tell 192.168.1.14
  1983 81.932397  Apple_a1:6e:48      IntelCor_16:96:ca   ARP      42 192.168.1.135 is at 52:e2:ef:ff:ff:f6

▷ Frame 1983: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▷ Ethernet II, Src: Apple_a1:6e:48 (d0:81:7a:a1:6e:48), Dst: IntelCor_16:96:ca (84:3a:4b:16:96:ca)
▷ Address Resolution Protocol (reply)

0000  84 3a 4b 16 96 ca d0 81  7a a1 6e 48 08 06 00 01   ·:K····· z·nH····
0010  08 00 06 04 00 02 52 e2  ef ff ff f6 c0 a8 01 87   ······R· ········
0020  84 3a 4b 16 96 ca c0 a8  01 0e                     ·:K····· ··

Ready to load or capture      Packets: 1986 · Displayed: 1792 (90.2%) · Dropped: 0 (0.0%)   Profile: Default
```

Fig. 11. Wireshark snapshot at Receiver site.

Further, in our algorithm one vital step is creation of the list of unallocated local IP addresses. If any ARP request or reply packet of an existing node gets lost over the network, it will result in an inaccurate list. Thus, to make sure we get accurate lists at the sender’s and the receiver’s site, the ARP requests for which no reply has been received in the first iteration, may be resent over the network. We called it as retries. Figure 12 and 13 shows the effect of increasing retries on time taken to create the unallocated list of local IP addresses and accuracy respectively. These values were calculated after taking an average of recorded values of error and time taken to create unallocated list for ten executions for each retrial value from 0 to 4. It was observed that the accuracy increased largely with one retry, but the increase in accuracy with further increase in the number of retries was minimal. Also, as we increased the number of retries the number of ARP packets over the network and execution time of our algorithm increased almost linearly. Thus, we freezed our algorithm with retry value of 1.

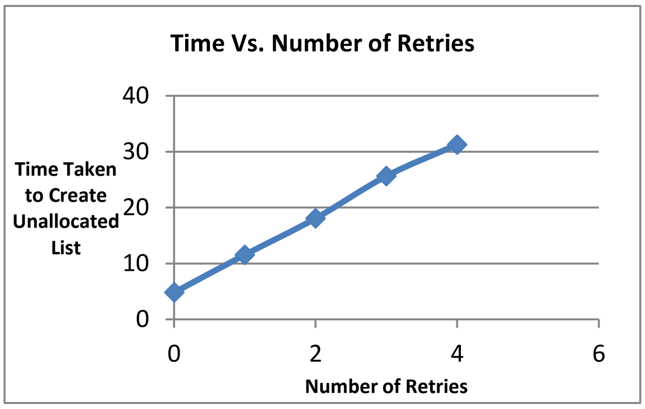


Fig. 12. Time Taken to Create Unallocated List Vs. Number of Retries

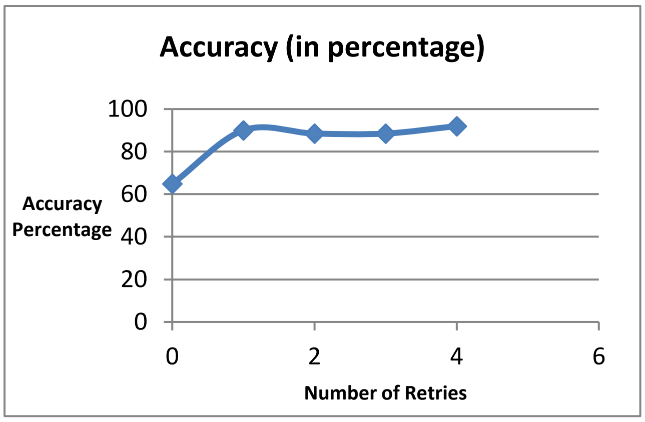


Fig. 13. Accuracy of unallocated list vs Number of Retries

VI. CONCLUSION

A Network Steganography technique ARPNetSteg, using Address Resolution protocol for Network Steganography is proposed and presented in this paper. In ARPNetSteg, the devices that wish to communicate covertly exploit the vulnerability of ARP spoofing partially without harming data flows to any other node on the network. Both the sender and receiver device start with creating a list of unallocated local IP addresses simultaneously. The sender and receiver use a common (mutually agreed) seed value to generate the same sequence of unused local IP addresses to communicate covertly. After receiving a desired ARP request as per the seed value, the sending device communicates by creating and sending a spoof ARP reply and further filling covert data and control information in the Sender Hardware Address field of this ARP reply. More ARP requests using the same seed are generated by the covert receiver after reading the control information present in the covert data last received. This control information carries the necessary information whether sender wants to send more covert data or not.

Our technique is capable of sending message of any length by breaking a large message into small messages of length 44 bits or lesser in each. Currently, the reliability of our technique depends upon matching of the lists of unallocated local IP addresses generated both at the sender's and the receiver's side. To make sure, that these lists match, we reconsidered all the unallocated local IP addresses present in the unallocated lists and generated Broadcast ARP request for all of them again. The local IP addresses for which a reply is received is further moved from unallocated to allocated list. With this, we successfully implemented a system that was able to send 44 bits of covert data per ARP reply message.

REFERENCES

[1] W. Richard Stevens. *TCP/IP illustrated (vol. 1): the protocols*. Addison-Wesley Longman Publishing Co., Inc., USA. 1993.

[2] Trithemius, Johannes, and Wolfgang Ernst Heidel. Steganographia. 1721.

[3] I. Cox, Miller, M., Bloom, J. Fridrich and T. Kalker. "*Digital Watermarking and Steganography*", 2nd ed. Elsevier, Morgan Kaufmann Publishers, 2008.

[4] K. Szczypiorski, "Steganography in TCP/IP networks", in *State of the Art and a Proposal of a New System–HICCUPS*, Institute of Telecommunications' seminar, Warsaw University of Technology, Poland, 2003.

[5] G. Fisk, M. Fisk, C. Papadopoulos and J. Neil, "Eliminating Steganography in Internet Traffic with Active Wardens," in *Proc. 5th International Workshop on Information Hiding*, Oct. 2002.

[6] D. Plummer, "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, Nov. 1982, DOI 10.17487/RFC0826.

[7] ARP, Address Resolution Protocol, "Network Socery Website," 2015 http://www.networksorcery.com/enp/protocol/arp.htm.

[8] TCP/IP Guide Website, 2020 http://www.tcpipguide.com/free/t_ARPMessageFormat.htm.

[9] T. G. Handel and M. T. Sandford, "Hiding data in the OSI network model." in *Proc. International Workshop on Information Hiding*, Berlin, Heidelberg, 1996, pp. 23-38.

[10] A. Mileva, and P. Boris, "Covert channels in TCP/IP protocol stack-extended version." *Open Computer Science* vol. 4, pp 45-66, 2014.

[11] C. Rowland, "Covert channels in the TCP/IP protocol suite, first Monday." *Peer Reviewed Journal on the Internet* vol. 2, no. 5, 1997.

[12] K. Ahsan, and D Kundur, "Practical data hiding in TCP/IP" in *Proc. Workshop on Multimedia Security at ACM Multimedia*, 2002.

[13] Bellovin, M. Steven, "Security problems in the TCP/IP protocol suite." *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 2, pp 32-48, 1989.

[14] K. Szczypiorski, M. Drzymała, and M. Ł. Urbański. "Network Steganography in the DNS Protocol." *International Journal of Electronics and Telecommunications*, vol. 62, no. 4, pp. 343-346, 2016.

[15] Z. Trabelsi and I. Jawhar, "Covert file transfer protocol based on the IP record route option." *Journal of Information Assurance and Security* vol. 5 no. 1, pp. 64-73, 2010.

[16] P. Bedi, A. Dua, "Network Steganography using the Overflow Field of Timestamp Option in an IPv4 Packet". Presented at Third International Conference on Computing and Network Communications (CoCoNet'19), Trivendrum, Dec. 18-21, 2019.

[17] L.Ji, Y.Fan, C.Ma, "Covert channel for local area network," in *Proc. IEEE International Conference on Wireless Communications, Networking and Information Security, WCNIS 2010,* Beijing, China, 2010, pp. 316–319.

[18] B. Jankowski, W. Mazurczyk, K. Szczypiorski, "PadSteg: Introducing Inter-Protocol Steganography," Telecommunication Systems, Vol. 52, 2013, pp. 1101–1111.

[19] S. Tobias, S. Wendzel, A. Mileva, and W. Mazurczyk, "Introducing dead drops to network steganography using ARP-caches and SNMP-walks," in *Proc. 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1-10.

[20] Scapy Website, 2020. https://scapy.readthedocs.io/en/latest/introduction.html

[21] Wireshark website, 2020, https://www.wireshark.org