Chapter

Analysis of Network Protocols: The Ability of Concealing the Information

Anton Noskov

Abstract

In this chapter, we consider the possibility of hidden data. Since today all network services rely on the basic protocols, the use of untestable and redundant fields may become a big problem. All of the modern data protocols have vulnerabilities. An attacker can use the reserved fields or field use undocumented way. Depending on the data transmission method and detection mechanisms, the technology for assessing the possibility of transmitting hidden information is changing. The work is of great practical interest for the implementation of systems to detect and prevent intrusions and data leaks in it. The authors determine the possibility of transmission and detection sends using a comparative evaluation of the fields in the packet with the values recommended in the standard protocol.

Keywords: network protocols, transport protocols, network analyze, network security

1. Introduction

Network steganography—type of steganography, in which secret data carriers use the network protocols of the OSI reference model—the open systems interconnection network model. In general, network steganography is a family of methods for modifying data in the headers of network protocols and in the payload fields of packets, changing the structure of packet transmission and hybrid methods in a particular network protocol (and sometimes several at once).

The transfer of hidden data in network steganography is carried out through hidden channels. The term "covert channel" introduced by Simmons in 1983 determined that the problem of information leakage is not limited to the use of software. A covert channel can exist in any open channel in which there is some redundancy. The hidden data is called steganogram. They are located in a specific carrier (carrier).

In network steganography, the role of the carrier is carried out by the packet transmitted over the network. The main parameters of network steganography are the bandwidth, covert channel, probability of detection, and steganographic cost. Bandwidth is the amount of secret data that can be sent per unit of time. The probability of detection is determined by the possibility of detecting a steganogram in a particular carrier. The most popular way to detect a steganogram is to analyze the statistical properties of the data obtained and compare them with typical values for this carrier. Steganographic cost characterizes the degree of change in the carrier after exposure to the steganographic method.

1.1 Network steganography methods

Baseline data for consideration classifications of methods and means of network steganography come from the materials of Polish scientists Mazurczyk and Szczypiorski and reports on the experiments of Canadian scientists Ahsan and Kundur, scientists Cauich and Gomez of the University of California at Irvine, and researchers Handel and Sandford at the National laboratory at Los Amos. All materials are freely available. Network steganography methods can be divided into three groups [1]:

- Steganography methods, whose essence is in changing data in the fields of the network protocol headers and in the packets payload fields.

- Steganography methods, in which the structure of packet transmission changes, for example, the sequence of packet transmission or the intentional introduction of packet loss during transmission.

- Mixed (hybrid) methods of steganography—when they are used, the contents of the packages, the delivery times of the packages, and the order of their transfer change.

Each of these methods is divided into several groups; for example, package modification methods include three different methods:

- Methods for changing data in protocol header fields: they are based on modifying the IP, Transmission Control Protocol (TCP), SCTP header fields, and so on.

- Packet payload modification methods; in this case, various watermark algorithms, speech codecs, and other steganographic techniques for hiding data are used.

- Methods of mixed techniques.

Methods for modifying the structure of gears and packages include three guidelines:

- Methods in which the order of the sequence of packets is changed.

- Methods that change the delay between packets.

- Methods, the essence of which is to introduce intentional packet loss by skipping sequence numbers at the sender.

Mixed (hybrid) methods of steganography use two approaches: methods of audio packet loss (LACK) [2] and packet retransmission (RSTEG) [1].

The main idea of methods for modifying header fields is to use some header fields to add steganogram to them [3, 4]. This is possible due to some redundancy in these fields, that is, there are certain conditions in which the values in these fields will not be used in the transmission of packets. The most commonly used header fields are IP and TCP protocols.

Consider an example of a similar method based on modifying unused IP protocol fields to create a hidden channel [4].

The value of the "Identification" field of the IP packet is generated to the sender side. This number contains a random number that is generated when a package is created. The "Identification" field is used only when fragmentation is used. Therefore, to use this method, you need to know the MTU value in the transmitted network and not exceed it, so that the packet is not fragmented during transmission. In the absence of the need for packet fragmentation, a certain redundancy occurs in the "Flags" field, in the second bit, which is responsible for setting the Don't Fragment (DF) flag. It is possible to specify a flag notifying the sender's unwillingness to fragment a packet. If the steganogram package is not fragmented due to its size, you can hide the information in the "DoNotFragmentBit" flag field. Using this method provides bandwidth of 1 bit.

The advantage of this method is the transmission of unchanged information from the sender to the recipient, but it also limits the amount of information sent. Steganography based on this method is easily implemented; has a good bandwidth, since you can send a lot of IP packets with the changes; and is low cost due to the use of fields that do not violate the functionality of the packet. Among the shortcomings it should be noted that the transmitted data is contained in the open form and can be easily read by the observer (although it is possible to strengthen the protection using additional cryptography).

Another method of modifying network packets that alters the payload of a VoIP packet can be widely used in practice with the popularity of programs that provide voice and video communications over the Internet. The network steganography method designed to hide VoIP messages is called Transcoding Steganography (TranSteg), a network steganography method that compresses the payload of a network packet by transcoding. TranSteg can be used in other applications or services (e.g., streaming video), where there is a possibility of compression (with or without losses) of open data. In TranSteg, data compression is used to make room for the steganogram: transcoding (lossy compression) of voice data from a high bitrate to a lower bitrate occurs with minimal loss of voice quality, and after compression, data is added to the free space in the payload package [5]. In general, the method allows to obtain more or less good steganographic bandwidth of 32 kb/s with the smallest difference in packet delay. Experiments of Polish scientists have shown that the delay in transmitting a VoIP packet using TranSteg increases by 1 ms, in contrast to a packet without a steganogram. The complexity of detection directly depends on the choice of the scenario and the conditions of the outside observer (e.g., its location). Among the shortcomings worth mentioning is the fact that this method is difficult to implement. It is necessary to find out which codecs the program uses for voice communication, to choose codecs with the smallest difference in speech quality, while giving more space for embedding steganograms. During compression, the quality of the transmitted speech information is lost.

Also interesting is the direction using the mechanisms of the SCTP protocol. Stream control transport protocol (SCTP) [6] is a packet-based transport protocol, a new-level transport protocol that will replace TCP and User Datagram Protocol (UDP) in future networks. Today, this protocol is implemented in operating systems such as BSD, Linux, HP-UX, and SunSolaris, supports network devices of the Cisco IOS operating system, and can be used in Windows. SCTP steganography uses new features of this protocol, such as multi-threading and the use of multiple interfaces (multi-homing).

The methods of SCTP steganography can be divided into three groups [7]:

- Methods in which the contents of SCTP packets change.
- Methods in which the sequence of transmission of SCTP packets is changed.
- Methods that affect both the content of packages and their order when transfer (hybrid method).

Methods for changing the contents of SCTP packets are based on the fact that each STCP packet is made up of parts and each of these parts can contain variable parameters. Regardless of the implementation, a statistical analysis of the addresses of the network cards used for the forwarded blocks can help in detecting hidden connections. Eliminating the possibility of applying this method, steganography can be achieved by changing the source and destination addresses in randomly selected packet, which is contained in the re-expel e PTO unit.

The essence of the hybrid method based on the SCTP protocol is to use certain protocol mechanisms that allow you to organize the intentional passing of packets in a stream without resending it. Later a steganogram is added to this packet, and it is resubmitted [7]. Modification of packages using a hybrid method can be presented on the Hidden Communication System for Corrupted Networks (HICCUPS), which uses the imperfections of data transmission in a network environment, such as interference and noise in a communication environment, as well as the usual susceptibility of data to distortion. HICCUPS is a steganographic system with bandwidth allocation in a public network environment. Wireless networks are more susceptible to data corruption than wired ones, so the use of noise and noise in the communication environment during system operation looks very tempting. "Listening" of all the frames with the transmitted data in the environment and the ability to send damaged frames with incorrectly corrected code values are two important network features necessary for the implementation of HICCUPS. In particular, wireless networks use an air connection with a variable bit error rate (BER), which makes it possible to introduce artificially damaged frames. This method has low bandwidth (network dependent), cumbersome implementation, low steganographic cost, and high detection complexity. However, the frame analysis does not involve checksum may lead to the discovery of the use of Nogo given method.

The RSTEG method is based on the packet resending mechanism, the essence of which is as follows: when the sender sends a packet, the recipient does not respond with a confirmation flag; thus the packet resending mechanism should work, and the packet with the steganogram inside will be sent again, but confirmation does not come. The next time this mechanism is triggered, the original packet is sent without hidden attachments, to which the packet arrives with confirmation of successful receipt.

The performance of an RSTEG depends on many factors, such as the details of the communication procedures (in particular, the size of the packet payload, the frequency with which segments are generated, and so on).

The investigated method of steganography using packet retransmission RSTEG is a hybrid. Therefore, its steganographic bandwidth is approximately equal to the bandwidth of the methods with packet modification and at the same time higher than the methods of changing the order of packet transmission. The complexity of detection and throughput is directly related to the use of the implementation mechanism of the method. RSTEG based on RTO is characterized by high detection complexity and low bandwidth, while SACK has the maximum bandwidth for RSTEG, but is also more easily detected. The use RSTEG utilizing TCP protocol is a good choice for IP networks. Among the shortcomings, it should be noted that this method is difficult to implement, especially its scenarios, which are based on interception and correction of packets transmitted by ordinary users. Due to the dramatically increased frequency of retransmitted packets or the unusual occurrence of delays in the transmission of steganograms, a casual observer may be suspicious.

Lost audio packets steganography (LACK)—steganography of deliberate delay of audio packets [2]. This is another method implemented via VoIP. Communication over IP telephony consists of two parts: signaling (dialing) and conversational. Both parts of the traffic are transmitted in both directions. The signaling protocols used are SIP and RTP (with RTCP acting as the control protocol). This means that during the signaling phase of the call, the SIP endpoints (called user SIP agents) exchange some SIP messages. Usually SIP messages pass through SIP servers: proxy or redirected, which allows users to search and find each other. After this stage, the conversation phase begins, where the audio (RTP) stream goes to both directions between the caller and the callee. This method has certain advantages. The bandwidth is not less and sometimes higher than the other algorithms that use audio packets. But if you intentionally cause losses, the quality of the connection deteriorates, which can become suspicious for both ordinary users and listeners. Based on the presented steganalysis LACK methods, it can be concluded that the method has an average detection complexity. The implementation of the method is too complex, but may not be possible within certain operating systems.

Table 1 shows a comparison of methods and their main characteristics and implementation. The position of each method in this table shows how much its characteristics are superior or inferior to the others. The higher the method displayed at the table, the more indicators of its characteristics. In the "Implementation" field, the simplicity of the organization of this method is considered. The less time and effort required by the implementation of this method, the higher its position in this title. Based on the data from **Table 1**, it can be concluded that the main characteristics are directly dependent on each other.

No	Throughput ability steganography	Complexity discoveries	Steganography cost	Implementation
1	TranSteg	HICCUPS	HICCUPS	Modification header fields TCP and IP packets
2	LACK	TranSteg	LACK	Modification data blocks in SCTP protocols
3	HICCUPS	LACK	RSTEG	TranSteg
4	RSTEG	RSTEG	TranSteg	Using SCTP multi-homing
5	Modification fields in TCP headers and IP packets	Using SCTP protocol (hybrid)	Protocol use SCTP (hybrid)	Using SCTP protocol (hybrid)
6	Modification data blocks in SCTP protocols	SCTP multi-homing	Modification of blocks data in SCTP protocols	LACK
7	Using SCTP protocol (hybrid)	Modification fields in TCP headers and IP packets	SCTP multi-homing	RSTEG
8	Using SCTP multi-homing	Modification data blocks in SCTP protocols	Modifying fields in TCP and IP headers packages	HICCUPS

Table 1.
Comparison of network steganography methods.

2. The combined method using modification of the fields IP and TCP

As mentioned earlier, the methods for modifying the IP and TCP header fields have certain features that make them stand out from the rest of the methods:

- The most common and standard protocols are used as carriers of the steganogram.
- Total gives bandwidth of 49 bits per 1 packet.
- Implemented on any operating system, the implementation does not require long adjustments and preparations.
- Changes in the package will not affect its behavior on the network, in case it will not be fragmented.

Despite the many advantages of both methods, there are some flaws, and the main one, to which attention is immediately drawn, is the obviousness of data transfer, i.e., any statistical analysis allows us to calculate both the hidden communication channel itself and the information transmitted in it.

The method proposed by Rowland [3] is as follows: to generate a value in the "Sequence Number" field, the plaintext character is encoded in accordance with the ASCII table, and the resulting value is multiplied by a certain number multiple of two. The resulting value is entered in the "Sequence Number" field and sent to the recipient. The recipient, knowing the key (divider), should check all incoming TCP packets for the subject of the steganogram, dividing the value of the "Sequence number" field by the key.

On the one hand, this method allows you to create a data channel through which you can transmit secret data in front of a passive observer. But the existence of a single key is a disadvantage, since, based on a dozen of such packages, it can be concluded that the sequence numbers of all packages have a common factor, which is the key. Thus, the proposed method is easy to detect.

Based on the source data and analysis of the disadvantages of network steganography methods with modification of the IP and TCP packet header fields, we can propose a modified method that will be based on the simultaneous use of the IP and TCP protocol header fields. The key needed to decrypt the transmitted message will also be transmitted as a steganogram, only in encrypted form in the "Identifier" field of the IP header, while the encrypted steganogram will be transmitted in the "Sequence number" field of the TCP header.

The implementation of this method is divided into two parts:

- Preparing data for the transfer, which includes generating the key k, converting the transmitted secret symbol or number into its corresponding code in the ASCII table, and calculating the value of the carrier C, which is an encrypted steganogram.
- Entering data into the corresponding TCP and IP header fields.

The first block consists of the following steps:

- Generation of the key k, which will be used in the future. The key can be any number that is a multiple of two. To generate a key, take two numbers x and y and raise the first to the power of the second.

- The conversion of secret data—a character or number that must be transferred to the corresponding code in the ASCII table. The coded number is denoted by S, since it is our steganogram.

- Getting the media C as the product of the key value by the value of a secret character.

$$C = S^{*}k_{10}$$

- Checking the number C—it must meet the requirement $2^{28} < C < 2^{33}$. This condition is necessary so that the value of the "Sequence number" field does not look suspicious. If the value of C does not meet the requirements, the numbers x and y need to be changed to others, and repeat steps 1–2. Further studies will be conducted on the automatic formation of x, y.

- The value of the numbers x and y is written together into the number z and is flipped so that the previous values can only be read from right to left.

Then the data is converted from decimal to hexadecimal. Thus, we get a three-digit hexadecimal number inv. (z) 16.

Then, at the second stage, you need to put the obtained values of the encrypted key and steganogram into the TCP and IP header fields.

We briefly describe the network steganography method with a modification of the fields in the TCP header, since in it we will transmit the secret message itself. For the purpose of steganography, the header of this protocol usually uses some fields that can be changed without losing the functionality of the package. For the purpose of our research, we will focus on the "Sequence Number" field (SN, SequenceNumber). This field performs two tasks. The first is the following: if the SYN flag is set, then this initial value of the sequence number is ISN (InitialSequenceNumber), and the first byte of data that will be transmitted in the next packet will have a sequence number equal to ISN + 1. Otherwise, if SYN is not set, the first byte of data transmitted in this packet has this sequence number. For our case it is important to know that this value will not change during the path of the packet from the sender to the recipient.

The "Sequence Number" field allows you to create a 32-bit length sequence. According to the Rowland method, the transmitted message is encoded in accordance with the ASCII table and multiplied by a certain number (the key), a multiple of two to reduce the detection probability, then entered into the generated TCP packet in the "Sequence number" field, and the packet is sent. When the packet reaches the destination address, the recipient must save all incoming TCP packets, from which he must remove the value in the "Sequence number" field and then divide by the key he knows in advance. But, as it was said before, this method is extremely easy to detect based on the analysis of a number of TCP packets due to a permanent key. In the proposed modification of the method, this key will be transmitted simultaneously with the TCP packet, in the IP header. This will increase the difficulty of detecting the steganogram.

The next step is to add the value of the C media in the "Sequence Number" field of the TCP header.

Next, you must enter the value of the encrypted key (inv (z)) 16 in the IP header field. To organize such an operation, you should return to the network steganography method with modification of the IP header fields. During the packet path, only the "Identifier" field remains unchanged; its length is 16 bits and 1 bit in the "Flags" field, which is responsible for the DF flag. Changing these fields does not carry changes in the package, in case the package is not fragmented, but it should not be, since by condition we need to know the minimum MTU value and not exceed it when creating and sending the package.

At the "Identifier" field, 16 bits is available to us for adding a steganogram; the information in it is displayed in the form of four numbers in hexadecimal number system. Thus, we have 65,535 possible values that can be used both for transmitting the steganogram and for the key, which in turn is also a steganogram. In order not to transmit the key in such an explicit form, it is proposed to use only three numbers out of four, while reading them from right to left. In this case, the number can be odd with its standard reading from left to right. The fourth unused number can take any value. Thus, we can use only 16 of the 17 bits available in a packet. It is proposed to use the second bit in the "Flags" field—DF—as a specific label, the presence of which allows you to expand the key extraction algorithm: whether you need to read the value from the first or from the second number in the "Identifier" field to extract the key.

Thus, the next step is to enter (inv (z)) 16 in the "Identifier" field of the IP header. At the same time, we must set the value of "1" to the second bit in the "Flags" field if we enter the key in the first 12 bytes of the "Identifier" field or 0 if we fill the first 4 bytes of the field with random values and in the remaining 12 bytes our key.

Next, we send a packet with modified fields to the recipient, where he must carry out the procedure inversely described in the framework of this algorithm [8].

We calculate the bandwidth of the proposed method.

Since the "Identifier" field in the IP header can contain 16 bits of information, 1 bit is available in the "Flags" field, and in the "Sequence number" field, a 32-bit information is available in the TCP header; we can conclude that the total throughput of steganography is 49 bits. But it should be noted that in this method we use the "Identifier" field to transmit the encrypted key in the steganogram, which is used to extract secret information from the "Sequence number" field, and the bit in the "Flags" field is used as a label. Thus, to transfer the encrypted key, we allocate 12 bits of information available in the "Identifier" field, and in the remaining 4 bits, we enter a random number from 0 to 16 in the hexadecimal number system (from 1 to F) and use 1 bit as a label, necessary for more organization more flexible operation of the algorithm. Based on this, we can conclude that for transmitting specific information, we have 32 bits left in the "Sequence number" field, and 3 bits of secret information can be transmitted, which is encrypted in 32 bits of information hiding the secret.

2.1 Intercomputer exchange

The exchange of computer networks is based on the Open System Interconnection (OSI) reference model.

Studying hidden information flows with computer interaction on networks of interest will include information about the services that are added to the network traffic data. As part of the protocol, headings are assessed at two levels: network and transport. We will address network protocols (IPv4 and IPv6) and transport protocols (TCP and UDP).

Further, we are considering the reports and the possibility of more detailed manipulation.

2.2 IPv4

IPv4 is the most popular protocol of network level; see more information in RFC791.

The header size of IPv4 is 20 bytes; using specialized field in header—"Options" field—can increase it. When the amount of the header is less than 20 bytes, it is likely damaged and has to be discarded.

2.3 Header of IPv4

The format of IPv4 header is presented in **Figure 1**.
IPv4 header field analysis shows the following results:

1. "Internet Header Length" field. Ability to increase the size of the Internet Header Length field to extend the original header. This change allows you to add data to the next two "Options" and "Padding" fields.

2. "Type of Service" field

   Bits from 0 to 2 are set for priority and 6 to 7 set to reserved.

   -0-2:

   The value "111" should not appear on the networks of provider; it could be appearing only for local networks, which leads to the point that the capture of this value in the network provider is a mark of malicious information injection.

   -6-7:

   By default, these bits are reserved and must be set to 0; the result is that the other value is possible injection information.

3. "Identification" field

   You can change the value of the identification field. The point is that the field is used to build correctly after fragmentation, but there is a DF flag that rejects fragment packets, so if the flag is set to "1" this ID is not required, and this field could be used to pass hidden information.

4. "Flags" field

   As the standard requires, the first bit is reserved and should be set to "0"; if the result is different, it is mark of injection information.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1.
Header of IPv4.

5. "Fragment Offset" field

   You can change the value of the "Fragment Offset" field. The best option is when the DF flag is set to "0," since the fragmentation strategy is designed so that an unfragmented datagram in all fields related to fragmentation has zero values. This means, despite the fact that the flag prevents fragmentation, we can still implement it in the offset of the fragment, but the fact of identification of the manipulation becomes more detectable.

6. "Source Address" field

   You can change the "Source Address" field value.

7. It should be noted that manipulation is possible only on the condition that the package consists of hidden source data. Since the manipulation will not be caused by the source of the information, the receiving site could not properly build the packets.

8. "Destination Address" field

   IPv4-in IPv6 headers can be encapsulated using the IPv4 Destination Address field to insert information into it. In this case, the IPv6 header will be responsible for delivering the package.

9. "Options" field

   The value of the options field is limited in the IPv4 header, and as a result of the analysis, we are trying to determine any field value that may appear in this type of field. So we may try to determine the incorrect significant of this field, the appearance of which indicates the possible malicious activity on the injection of information.

10. "Padding" field

   This field goes after value 0x00 of the "Options" field; the value is the EOL and takes up to 32-bit header boundaries. The interest in this manipulation is that after the optional EOL, the equipment does not examine headers on 32-bit boundaries; this means that these bytes are invisible to network devices and sniffer. Although the analysis of this field is simple enough, the EOL up to 32-bit header boundaries must be set to "0" at the standard behind the "Options" field, causing any other value of this field to indicate that the data is being injected.

2.4 Injection's result

The standard IPv4 header size with options and fields with padding is 320 bits. Two different options need to be considered:

1. IPv4 is a carrier and is responsible for packet addressing. Due to manipulation, 182 bits can be used, which is 56.88% of the total number of bits. This volume allows you to insert 22 symbols from 8 bits in ASCII encoding into the header.

So after calculations we have got a value up to 4 bits. This remainder is part of the other 8 bits of the transmitted information.

2. IPv4 is a passenger, it's an IPv4 encapsulated header in other headers, such as IPv6 or GRE. In this case, the method for implementing the target address can be used. As a result, handling bits 214, 66% of the total number of bits can be used. This volume allows you to implement a 26-character header with 8 bits in ASCII encoding. Thus, after calculations, a value of 6 bits is obtained. The treated residue was included in an additional 8 bits of the transmitted symbol.

2.5 IPv6

2.5.1 Header of IPv6

The header's format of IPv6 is presented in **Figure 2**.

1. "Traffic Class" field

   You can change the "Traffic Class" value arbitrarily. This manipulation cannot be detected by analysis.

2. "Flow description" field

   You can change the value of the "Flow Label" field.

   This manipulation cannot be detected by the packet sniffer.

3. "Load Length" field

   It is possible to increase the size of this field when adding data to the end of the original IP packet, like IPv4. This modification cannot be detected by the packet sniffer.

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version| Traffic Class |           Flow Label                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Payload Length        |  Next Header  |   Hop Limit   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                         Source Address                        +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                      Destination Address                      +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.
Header format of IPv6.

4. “Source Address” field

   You have the possibility to change the data of this field at IPv4 format, but international standards from the IPv6 community do not recommend using it as a source address.

5. “Destination Address” field

   In this protocol, you can use the IPv6 “Destination Address” field in the IPv4 encapsulation header to load information into it. In this case, the IPv4 header will be responsible for the packet delivery.

   This manipulation cannot be detected by the packet sniffer.

3. Result of injection

The standard IPv6 header size with options and fields with padding is 320 bits. Two different options need to be considered:

1. IPv6 is a carrier, that is, it is responsible for addressing the package. As a result of the manipulations described above, 156 bits can be used, which is 48.75% of the total number of bits. This volume allows you to insert a caption with 19 characters from 8 bits into the ASCII character set. Thus, after calculations get a value of 4 bits. The treated residue was included in an additional 8 bits of the transmitted symbol.

2. IPv6 is a passenger and is transmitted by IPv6 encapsulation header to other headers, such as IPv4 or GRE. In this case, the method for implementing the target address can be used. As a result of the manipulations described above, it is possible to use 284 bits, which is 88.75% of the total number of bits. This volume allows you to implement a 35-character header with 8 bits in ASCII. Thus, after calculations, we get a possible value of 4 bits. The processed remainder will be added as an additional 8 bits of transmitted characters.

3.1 TCP

Transmission Control Protocol is a reliable protocol of transport layer. TCP is oriented to establish a logical connection, that is, the hosts negotiate and create a session and then begin to transfer data. Every time a package is sent, the sender is awaiting acknowledgement of delivery receipt. This protocol is standardized by RFC 793.

3.1.1 Header of TCP

Header’s format of TCP is presented in **Figure 3**.
“Source Port” field

1. You can change the “Source Port” field value. Processing is only possible when the package was a hidden data source. Due to the manipulations that occur on the source host, the receiving party will not be able to properly assemble the original packet. This manipulation cannot be detected by the packet sniffer.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3.
Header format TCP.

2. "Destination Port" field

   You can change the value of "Destination Port" field. Handling is only possible when the packet was a hidden data source. Due to the manipulations that occur on the source host, the receiving party will not be able to properly assemble the original packet. This modification cannot be detected by sniffer.

3. "Sequence Number" field

   We could modify the information in this field. Processing is only possible when the package was a source of hidden data. Because of the modification that had occurred at the source device, the receiving PC cannot correctly build the original packets. This modification cannot be observed by the network sniffer.

4. "Acknowledgment Number" field

   We could change the contents of this field. Modification is allowed provided that the package was made up source of hidden data. Due to the modification that occurred on the source host, the receiving party will not be able to properly assemble the original packet.

5. "Data Offset" field

   The manipulation is as follows: this increases the size of the "Data Offset" field, expands the TCP header, and adds a parameter field. In the options you can add data after byte 0x00 EOL.

   At standard byte 0x00 EOL, bytes with a value of "0" should be due to some other value that indicates that a data injection has occurred.

6. "Reserved" field

   You can modify the value of this field.

   By default, the values of all standard bits must be set to "0" as a result of some other values that indicate that a data injection has been occurred.

7. "Window" field

   You can modify the value of the "Window" field. Handling is only possible when the packet was built at a hidden data source. Due to the manipulations that occur on the source host, the receiving party will not be able to properly assemble the original packet

8. "Pointer Urgent" field

   You can modify the value of this field. This injection is only possible if all URG options are present.

   So, if the Urgent Pointer is filled in and the flag of URG is not setting, it means that the Urgent Pointer is not used correctly.

9. "Options" field

We could modify the data of this field. In the options, you can realize the data after value 0x00, but it is not considered after this byte header data.

TCP header option values are limited, and network analysis results in attempting to identify a possible option that attempts to identify incorrectly filled options or unknown options whose appearance indicates a possible injection of information.

10. "Padding" field

    It is possible to fill the field of any padding.

    It should be noted that manipulation is only possible if the package is made up of hidden source data. Because of the manipulation that occurs at the source, the receiving party cannot properly collect packets.

Handling "Padding" is one of the most interesting. The "Padding" field starts after the 0x00 in the "Options" field; the value is the EOL option and takes up to 32-bit header boundaries. Interest in this manipulation is contained in the following text after the EOL does not produce a 32-bit header, which means that these bytes are invisible to network devices and sniffer. Although the analysis of this field is simple enough, the EOL up to 32-bit header boundaries must be set to "0" at the standard behind the "Options" field, causing any other value of this field to indicate that the data is being injected.

4. Result of injection

The standard TCP header field with options and fall is 192 bits. As a result of the above actions, you can use up to 150 bits, which is 78.13% of the total number of bits in the original, unmodified header. This amount of data allows the use of 18 characters in an 8-bit header in the standard ASCII character set. Therefore, after all the calculations, we get the maximum possible amount equal to 6 bits. The processed piece of information was included in the next 8 bits of the transmitted symbol.

DOI: http://dx.doi.org/10.5772/intechopen.88098

4.1 UDP

User Datagram Protocol is a connectionless transport layer protocol. No connection setup is created before transferring between hosts. This protocol is less reliable than TCP, but gives a higher transfer rate with less overhead. This protocol is standardized by RFC 768.

4.1.1 Header field of UDP

Header's format of UDP is presented in **Figure 4**.
"Source Port" field

1. You can change the "Source Port" field value. Processing is only possible when the package was a hidden data source. Because of the manipulation that had occurred at the source device, the receiving party cannot properly assemble the original packets. This manipulation cannot be detected by the packet sniffer.

2. "Destination Port" field

   You can change the "Target Port" field value. Processing is only possible when the package was created by a hidden data source. Due to the manipulation of the device generating the packages, the receiving party cannot correctly assemble the source packages. This modification cannot be detected by the network sniffer.

3. "Length" field

   We could change the significance of the "Length" field. Increasing the value of this field has also increased the size of the package, so we can change the fields of data octets by appending to the end of datagram.

So processing is possible when the package was a source of hidden data. So if the modification had occurred at the source device, the receiving host cannot properly assemble the original packets. This manipulation cannot be detected by the packet sniffer.

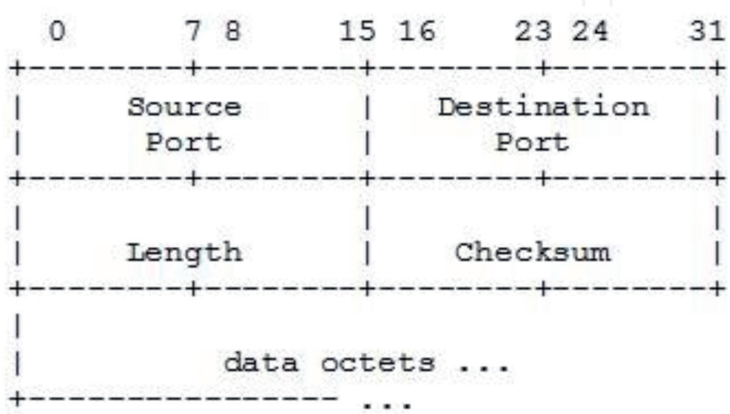

Figure 4.
Header of UDP.

5. Result of injection

The size of the UDP header of the datagram is 64 bits; as a result of the described changes, you can use 32 bits, which is 50% of the total number of bits in the header, which allows you to implement a 4–8-bit header in the ASCII character set.

6. Conclusion

In this work, we began to develop methods and special software for generating bitstreams in order to organize a secure connection.

This software method was implemented in software, ensuring secure network communication. The main part of the program model is a detector program for analyzing network traffic to search for possible hidden transmissions. The analysis is implemented by checking the header in compliance with the standards, which is needed to identify unauthorized values for specific areas of the PDU.

The surveys revealed possible vulnerabilities that could embed relevant information in the puncture headers we reviewed. **Table 2** presents the quantification of the study results, showing the remainder is the number of bits that are part of the next 8 bits of the transmitted symbol.

It should be noted that TCP was created as a reliable protocol for delivery, but after entering the hidden data by the TCP header proposed above, changes made to the header fields of the TCP lead to the loss of the functionality of a reliable protocol, making it similar to the UDP.

In the created model, the transmission of one packet is realized, that is, the full message is embedded in all possible of headers at only one datagram. In order to see the maximum feasible messaging, we chose the following protocols: IPv4, IPv6, and TCP. Note that for simplicity, TCP header data is not included in the fragment offset. Thus, thanks to the proposed manipulation, the programming model uses 603 bits, which is 74.04% of the total number of bits in the order of three headers. This volume allows you to enter 75 characters out of 8 bits in ASCII encoding.

Protocol	Size of injection information (bits)	Percentage of the total header size (%)	The number of symbols	Rest bits
IPv4 (carrier)	182	56.88	22	6
IPv4 (passenger)	214	66	26	6
IPv6 (carrier)	156	48.75	19	4
IPv6 (passenger)	284	88.75	35	4
TCP	150	78.13	18	6
UDP	32	50	4	0

Table 2.
The quantification of the study results.

Author details

Anton Noskov
Yaroslavl State Technical University, Yaroslavl, Russia

*Address all correspondence to: anton.noskov@gmail.com

References

[1] Mazurczyk W, Szczypiorski K. In: Meersman R, Tari Z, editors. Steganography of VoIP Streams. Springer-Verlag; 2009. Available from: http://home.elka.pw.edu.pl/~wmazurcz/moja/art/OTM_StegVoIP_2008.pdf

[2] Mazurczyk W, Szaga P, Szczypiorski K. Retransmission Steganography and Its Detection. Available from: http://cygnus.tele.pw.edu.pl/~wmazurczyk/art/RSTEG.pdf

[3] Rowland CH. Covert channels in the TCP/IP protocol suite. Central European Journal of Computer Science. 1997;**2**(5):45-66. Available from: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/528/449

[4] Cauich E, Gómez R, Watanabe R. Data Hiding in Identification and Offset IP fields. California University at Irwing, Computer Science and Engineering. Irvine, CA, USA: University of California. http://www.sciweavers.org/read/data-hiding-in-identification-and-offset-ip-fields-124683

[5] Mazurczyk W, Szaga P, Szczypiorski K. Using Transcoding for Hidden Communication in IP Telephony. Warsaw University of Technology, Institute of Telecommunications; 2011. Available from: http://arxiv.org/pdf/1111.1250v1.pdf

[6] Stewart R. ed. Stream Control Transmission Protocol. – RFC 4960:6. Request for Comments: 4960, 2007. Available from: http://tools.ietf.org/html/rfc4960

[7] Frączek W, Mazurczyk W, Szczypiorski K. Stream Control Transmission Protocol Steganography. Warsaw University of Technology, Institute of Telecommunications; 2010. Available from: http://arxiv.org/abs/1006.0247

[8] ISO9646. Open System Interconnection, Conformance Testing Methodology and Framework. Switzerland, 1992