Pak. j. life soc. Sci. (2025), 23(2): 238-248 E-ISSN: 2221-7630;P-ISSN: 1727-4915

Pakistan Journal of Life and Social Sciences

www.pjlss.edu.pk

https://doi.org/10.57239/PJLSS-2025-23.2.00161

RESEARCH ARTICLE

Development of Methods and Tools for Implementing and Detecting Network Steganography

Elena Revyakina[1], Andrey Gazizov[2]

[1,2]Don State Technical University, Rostov-on-Don, Russia

ARTICLE INFO

Received: May 3, 2025

Accepted: July 15, 2025

Keywords

Traffic Analysis
Attack
Network Packets
Information Safety
Active Intelligence Methods

***Corresponding Author:**

elena.a.revyakina@gmail.com

ABSTRACT

The increasing sophistication of cyber threats has led to the exploitation of network steganography, that enables covert data transmission by embedding hidden information within standard network protocols such as ICMP and DNS. Traditional security mechanisms often fail to detect these transmissions, making network steganography a significant risk for data breaches, cyber espionage, and unauthorized communications. The purpose of the article was to develop and implement methods and tools for both embedding and detecting steganographic messages in network traffic; identify hidden data transmission vulnerabilities within service network protocols and propose efficient detection mechanisms that can help mitigate these security risks. A custom-built client-server application was developed using packet generation tools such as WinPcap, SharpPcap, and Packet.NET to analyze the feasibility of embedding and detecting hidden messages in ICMP packets. Various detection techniques, including statistical traffic analysis, machine learning models, and visual anomaly detection, were employed to identify deviations in network behavior. The effectiveness of these techniques was evaluated through controlled experiments. The study found that network steganography alters traffic characteristics in measurable ways, making detection possible. Key indicators of steganographic communication include increased frequency of service packets, unusual packet size distributions, and deviations in traffic flow patterns. The integration of machine learning models significantly improved the accuracy of detecting steganographic transmissions, surpassing traditional statistical methods. The findings demonstrate that steganography detection is achievable through statistical and AI-based analysis. Additionally, the study highlights the dual-use nature of network steganography, emphasizing its potential for both malicious exploitation and secure communications in defense, intelligence, and business applications.

INTRODUCTION

Today, in parallel with constantly developing technologies and means of data transmission, new opportunities are emerging for attackers to circumvent traditional security methods. One of these methods is provided by network steganography, which can be used to quietly embed the necessary information in the data stream and then extract it.

In the modern digital age, maintaining the confidentiality and security of network communications is a priority for many organizations and States. The fundamental requirement for data transmission is to prevent the transmission of hidden information blocks in network traffic, which can escape attention and not be filtered by network screens and other security systems. This requirement is dictated by the need to exclude the possibility of unauthorized access to information or its distortion.

However, the implementation of this requirement faces the problem of complexity and diversity of modern network protocols. Within these protocols, there are many potential places where data can be hidden or modified without directly affecting the visible parameters of the packet.

To successfully detect and counter such unauthorized actions, it is important to have a good understanding of the structure of data packets and the basic principles of network interaction. Only a detailed knowledge of the mechanisms of communication protocols can make it possible to create methods aimed at detecting anomalies in network traffic (Belkina, 2018).

In this regard, the task of developing methods and approaches that would be sensitive to minimal changes in network traffic parameters resulting from attempts to modify standard packets or interfere with the normal operation of communication protocols becomes urgent. Their timely detection becomes an integral part of ensuring reliable and secure operation of network systems.

It should also be noted that network steganography can be used not only for malicious purposes, but also to create hidden and secure communication channels in various areas, including defense, intelligence, and business. This leads to the need to study not only methods for detecting information embedded in network traffic, but also the formation of hidden steganographic communication channels (Ganzhur et al., 2018).

The purpose of this study is to study and implement methods and tools for implementing and identifying network steganography. To achieve this goal, we analyzed the technology of telecommunications systems and identified potential opportunities for data integration into network packages. A system for data exchange via service protocols has been developed. As a result, we proposed methods for detecting hidden data in network traffic using the developed system, as well as traffic analysis tools, and conduct an experimental study on the detection of steganographic packets.

MATERIALS AND METHODS

The main principle of organizing communication in computer networks is the use of standardized protocols and models to ensure the interaction of various devices. To understand the structure and operation of modern networks, it is extremely important to become familiar with the OSI model and basic concepts such as protocol, packet, and encapsulation.

And encapsulation in computer networks is the process of adding a header (and possibly a terminating block) to data at each layer of the OSI or TCP/IP model. When data is transferred from a higher layer to a lower one (for example, from the application layer to the transport layer), headers of the corresponding layer are added to this data. These headers contain service information that is necessary for data processing at each stage.

After adding a header at one level, the combination of data and a new header is passed to the next level,and the deprocess is,repeated. This process continues until the packet reaches the physical layer and is sent over the physical communication channel (Golubev et al., 2019).

In the context of network steganography, encapsulation plays a key role, as the original packet is "packed" into a new packet for transmission over another network. Thus, the original data becomes the "payload" for the new packet. When this new packet reaches its destination in the tunnels, it is decapsulated and the original packet is routed to its final destination.

To detect network steganography, various methods can be used: statistical analysis; comparison with a sample; search for specific signatures; machine learning; visual analysis. Statistical analysis is based on the search for changes in data packets that may cause anomalies in the statistical characteristics of network traffic. For example, hidden data transmission may affect packet size distribution or latency.

If you know what "clean" traffic looks like without steganography, you can compare it with current traffic to identify unusual deviations that may indicate the use of steganography. This method is called sample comparison. The search for specific signatures is used for those steganography methods that have unique characteristics and can be identified using signature detection methods. Machine learning is one of the promising methods for solving many complex problems. In relation to the problem discussed in this paper, you can train machine models to recognize signs of

steganography in network traffic, using examples of known steganographic traffic. In some cases, experts can manually perform visual analysis of network traffic for anomalies that may indicate the presence of steganography.

Knowing about possible steganography techniques, a network administrator can try to disrupt or modify potentially hidden data by adding noise or changing the packet structure.

The comparison process with the sample includes the following steps:

1. Collecting a reference sample. First, you need to collect or define a traffic reference sample. This may be a sample of traffic from your own network at a time when you can definitely say that there is no steganography, or it may be a generally accepted standard of "normal" traffic.

2. Extract characteristics. Certain characteristics are extracted from both datasets (current traffic and reference sample), such as packet sizes, time intervals between packets, header characteristics, and others.

3. Comparison of characteristics. These characteristics are then compared with each other. If significant differences are found between the current traffic and the reference sample, this may indicate the presence of steganography.

4. Limits and thresholds. Thresholds can be set to automate the process. If the difference in certain characteristics exceeds these limits, the system can automatically warn about the possible presence of steganography.

5. Additional tests. If suspicious deviations are detected, they can be further checked using other detection methods to refine the results.

Using machine learning to detect network steganography is an advanced approach that can recognize complex patterns and anomalies in network traffic that may not be visible to traditional detection methods.

Machine learning offers a dynamic and flexible way to detect network steganography, enabling systems to adapt to new and changing threats. However, this approach also requires deep expertise in machine learning, high-quality training data, and thorough validation (Pavlin et al., 2014).

Visual analysis for detecting network steganography is an approach in which graphical or visualized representations of network traffic are used to identify anomalies or unusual patterns that may indicate the presence of steganography. This method often relies on an expert's intuitive understanding and experience.

The first step in this process is data transformation, during which raw network traffic data is converted to a format suitable for visualization. This may include converting time series, statistics, or even the contents of packages into graphical images.

There are many tools and software that specialize in visualizing network traffic. These tools can present graphics in video graphs, dim maps, scatter charts, and other visual formats. The Expert Advisor searches for unusual or abnormal patterns in the visualized data. This may include unusual peaks in activity, uncharacteristic distributions, or other visual features that stand out from the usual traffic pattern. If the expert has access to samples of known steganographic traffic, they can compare the current visualization with these samples to determine similarities.

Some visualization tools allow users to interact with data by going into more detail or changing the visualization settings for better understanding.

Visual analysis relies on the human ability to recognize patterns, which makes this method subjective. In addition, this approach may be less effective when analyzing large amounts of data or when steganographic signals are thin and difficult to distinguish.

Visual analysis can be combined with other detection methods, such as statistical analysis or machine learning, to increase the accuracy of detection.

Tools for Generating and Analyzing Packets

To obtain detailed information about the functioning of a particular computer network, as well as analyze the data transmitted over it, a wide range of software utilities is used, including sniffers, port scanners, packet generators, and other specialized programs (Peskova et al., 2017).

The practical implementation of the ICMPencapsulation method described earlier is possible using the listed tools and is reduced to two tasks:

1. Forming a packet with the required content in the header and payload fields, which differs from the standard packets generated by the operating system.

2. Selection of the generated packet from the general traffic flow by the receiving party and recognition of the data placed in it.

To solve the first problem, you need to use a packet generator – a utility that allows you to set the contents of each field in the packet and send it once or at a certain interval, and to solve the second problem — a sniffer. Figure 1 shows an example of generating an ICMPpacket, during which the destination IPaddress, code, ICMPmessage type, and data were manually specified, and the remaining fields that need to be calculated were filled in automatically based on the entered values.

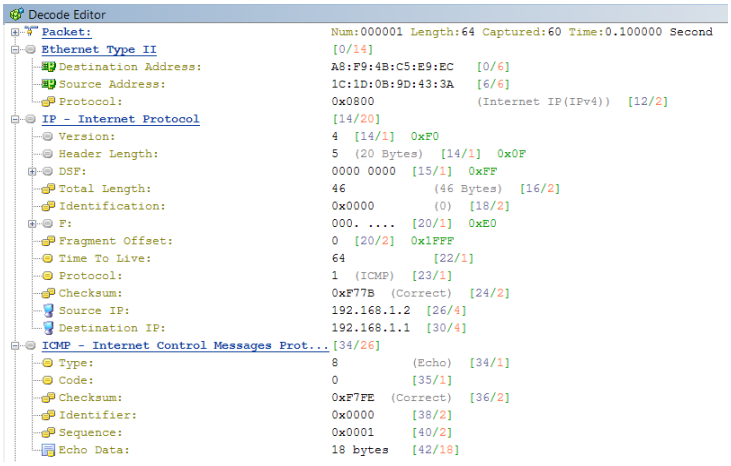

Figure 1. Creating a Package in ColasoftPacket Builder

As a sniffer for receiving and recognizing sent packets, you can use the Wireshark utility, which shows detailed information about all network packets passing through the computer. The result of its operation is shown in Figure 2. In the figure, among other packets, you can see an ICMP request in the "Data" field, which contains the "test message"test.

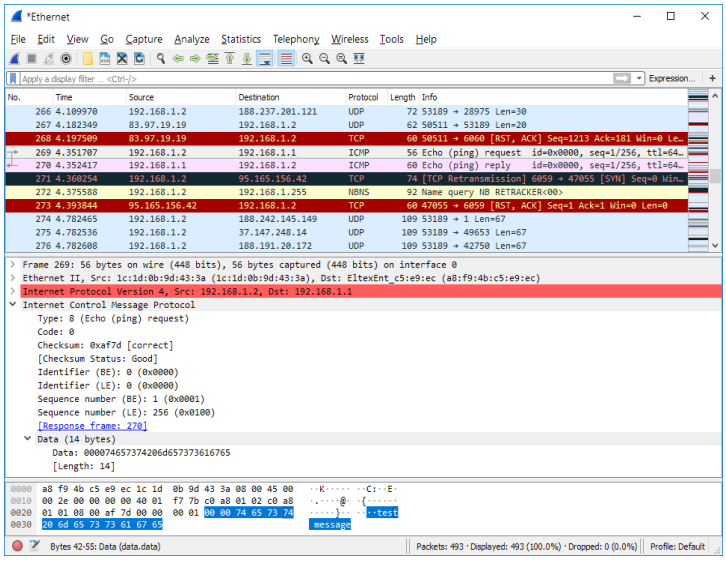

Figure 2. Receiving a Packet in Wireshark

Software Implementation of the Network Steganography Method

The method of interaction presented above, due to the need to manually fill in all fields of the packet headers, cannot provide efficient transmission of the required data volumes and is more suitable for testing purposes. For practical application of ICMP steganography, it is necessary to develop your own client-server application, in which the described functions for generating, sending, receiving, and recognizing packets will be performed automatically. The network socket interface discussed earlier is suitable for network communication at the upper levels of the OSI model, in particular, based on the DNS protocol. However, a significant modification of the packet exchange rules used in the proposed steganography methods is not possible using standard approaches to implementing client-server information systems and requires more control over the created and received packets. Such features are provided by the software interface of the so-called Raw sockets.

However, the correct implementation of a large number of protocols can be difficult due to the need to take into account a large number of requirements in different protocols, and therefore the preferred option is to use additional libraries of software components that work on the basis of raw sockets, but are easier to use. In this paper, we used SharpPcap libraries SharpPCap, based on WinPcap, and Packet.NET.

WinPcapis a low-level library for Windows operating systemsthat allows applications to interact with network interface drivers, as well as capture and transmit network packets bypassing the protocol stack. WinPcap consists of a driver that extends the operating system to support low-level hardware access, and a library that directly provides an interface to applications. The Wireshark utility mentioned earlier Wiresharkis also based on the WinPcap library (Malyuk, 2004).

SharpPcap allows you to capture all traffic passing through the selected computer interface for further analysis and processing in your own program. It supports most common protocols, and for unsupported protocols, it is possible to get data from the package in binary form for further program analysis. Packet.NET In turn, it provides software tools for generating its own packages of one of the protocols it supports. Figure 3 shows the interaction scheme of the considered software environments with the network adapter of the computer, as well as the standard method of application interaction through the implementation of the TCP/IP protocol stack in the operating system.

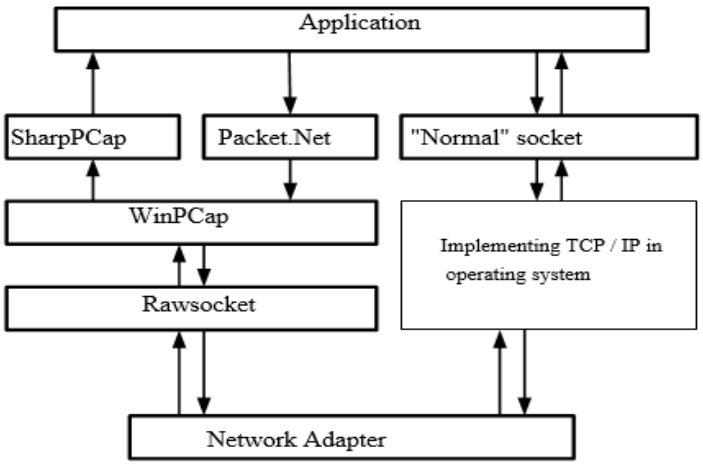

Figure 3: Network Adapter Interaction Diagram

Usage Packet.NET it consists in writing an algorithm for sequentially generating packets at different levels of the TCP/IP model, from the upper (application) to the lower (media access level). Accordingly, the software implementation of the steganography method under consideration includes several stages of encapsulation (Sorokin, 2020).

The algorithm of the program's operation when sending a message is reduced to the sequential formation of the final Ethernet packet, which will include an IP and ICMP packet with correctly filled header fields.

At the initial stage, an ICMP packet is created, which, in accordance with the object-oriented programming paradigm, is represented by an object of the ICMPv4Packet class described in the library Packet.NET. You must set the values of the header fields described below.

Type Code — numeric identifier of the message type. In the library Packet.NET The values for this field are stored in the ICMPv4TypeCodes enumeration and allow you to use their more understandable text equivalents instead of numbers. When a value is selected, the "type" and, if necessary, "code" fields in the package header are automatically filled in. These fields must be filled in correctly only if the ICMP protocol is used for its intended purpose, and any values can be used to generate an arbitrary packet, including the previously mentioned Echo Request or Echo Reply values.

The Sequence and ID fields in the packet's header are required to identify it in tasks that require sending multiple packets. This field must be used when sending a long message, which is divided into several shorter ones, each of which is placed in a separate ICMP packet with its own number in the sequence specified by an integer that starts with 0 and increases by 1 for each subsequent packet, up to the value216-1-1 (Shcheglov, 2009).

Payload Data — field for recording service data. However, when implementing the described system, it is in it that the information that needs to be transmitted through a hidden channel will be recorded.

RESULTS

After that, the created packet is transmitted to the underlying network layer protocol — IP, which requires filling in the source Address and destination Address fields intended for recording the IP addresses of the sender and recipient. The sender's address can be obtained from the properties of the network interface, and the recipient's address must be set by the user. The packet is then transmitted to the next channel layer.

At the link layer, just like the previous one, you need to add the source and destination MAC addresses to the packet header. At the same time, the source MAC address, as well as its IP address, can be obtained from the properties of the network interface, and to determine the destination MAC address, you can use various methods, such as an ARP table, sending an ARP request, or entering it manually. This package also requires calculating the checksum using the Update Checksum methodUpdateChecksum. After successful completion of all the described actions, the Ethernet packet generated at the link layer can be transmitted to the network.

The receiving party must perform the same steps, but in reverse order, i.e. first receive the Ethernet packet, extract the IP packet from it, then extract the ICMP packet from the IP packet and read the data recorded in it. To do this, the following decapsulation functions from the Sharp cap library are consistently used:

Ethernet Packet. Get Encapsulated(packet)

IPv4Packet.GetEncapsulated(packet)

ICMPv4Packet.GetEncapsulated(packet)

In the listed functions, "packet" is a parameter that is passed as a network packet that came from the previous layer of the OSI model. The result of the last function is an object that represents the structure and content of an ICMP packet, the "Data" field of which contains information transmitted through the tunnel.

After the libraries ' functioning principle was analyzed Packet.NET and Sharp cap was created as a test app. It creates an ICMP packet with the required content and sends it to the specified IP and MAC address via the selected network interface (Figure 4).

A network interface is a software representation of a hardware device that allows operating systems and applications to transmit data over a computer network (Frączek at el., 2016).

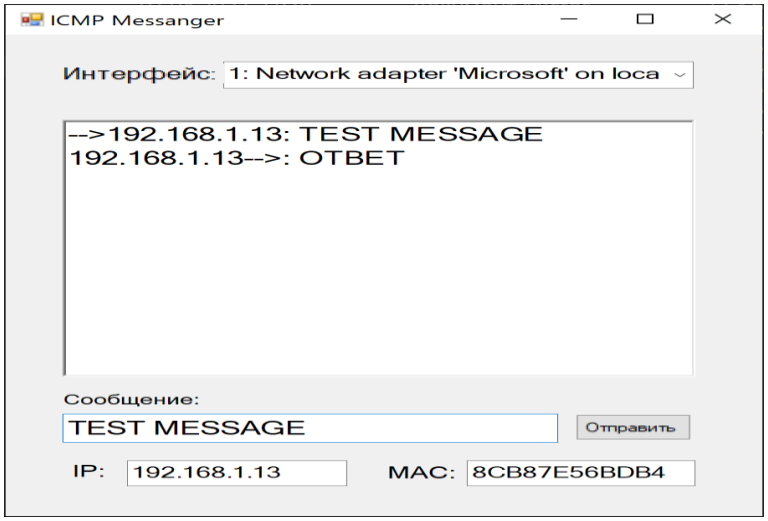

Figure 4: Main Application Form

In the figure, the "interface" input field is used to select one of the available network adapters on your computer. An ICMP packet will be sent through it. Of course, the selected network interface must be connected to some network, otherwise the message will not be sent. The adapter selected in the example figure is called "Network adapter Microsoftonlocalhost". In fact, this device is a built-in network card of the computer (Aknin, 2000).

The "MAC" field is used to record the MAC address of the recipient's network interface. Recipient's MAC address: "8C:B8:7E:56:BD: B4".

The "IP" field is used to record the IP address of the recipient of the ICMP packet. Recipient's IP address: "192.168.1.13".

The "Message" field records the data that needs to be sent to the recipient. In this case, the information is: "TESTMESSAGE".

After all the fields are filled in, click on the "Submit" button. After that, an ICMP packet with type 8 is generated. Using encapsulation, the IP and MAC address are added to it, and then this packet is sent to the recipient. In the second instance of the application, you can detect the sent message and send a response.

In order, to see not only the message, but also all the information about the ICMP packet, use the Wireshark utility. When sending a message, the traffic analyzer "catches" the ICMP packet (Figure 5). In this figure, you can see the IP address and MAC address of the sender and recipient, the type of ICMP packet, and the message text itself.

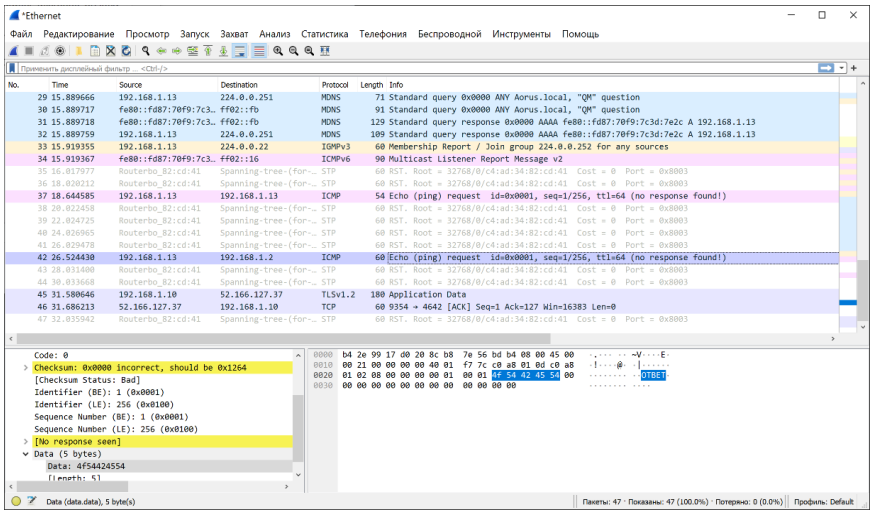

Figure 5: Detailed Information about the Packet in Wireshark5

All previous actions were performed when the firewall and anti-virus protection were disabled, as theoretically this could make it difficult to transmit the message (Galatenko, 2006).

In order, to check the effectiveness of programs, in other conditions, you need to run the firewall and antivirus protection. The test computer has ESETInternetSecurity antivirus installed, which manages all firewall parameters. Accordingly, to enable Firewall and antivirus protection, go to the ESETInternetSecurity settings and run the necessary parameters (Figure 6).

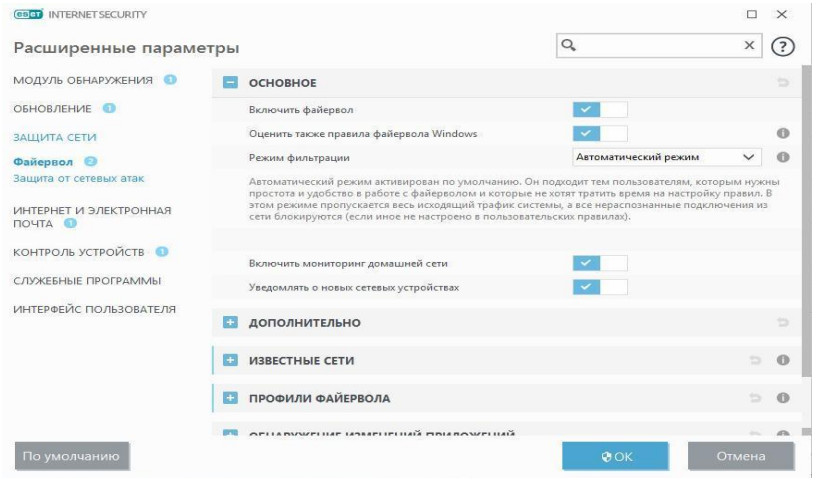

Figure 6: Launching the Network Screen

After starting, you can repeat the procedure for sending a message, which will still be delivered to the recipient. This is due to the fact that the firewall acts on the transport layer of the TCP/IP protocol stack, and the ICMP protocol works on the network layer, and the created ICMP packet with type 8 is actually an echo request, and does not fall under the standard firewall rules.

As a result, the resulting software implementation of the ICMP steganography method is capable of performing hidden data transmission when blocking transport layer protocols (Cherckesova et al., 2024).

Statistical Analysis of Traffic to Identify Steganographic Messages

Statistical traffic analysis uses various parameters and characteristics to search for steganographic messages, which can help identify anomalies or hidden data in network traffic. These parameters include:

- distribution of packet sizes.

- time intervals between packets.

- distribution of bits and bytes in packets.

- frequency characteristics.

- correlations between packages.

- specific network protocols.

Of these methods, frequency response analysis and comparison with normal traffic are suitable for detecting ICMP steganography. To implement these methods, we use the Wireshark packet sniffer discussed earlier Wireshark. Statistical analysis in Wireshark is one of the many features of this tool and can be used to identify patterns that may indicate steganography or other anomalies in network traffic. Statistical analysis functions include calculating statistics based on protocols, packet sizes, sources and destinations, time intervals, and data transmission errors (Kodatsky et al., 2024).

Protocol statistics allow comparison with normal traffic. To do this, the experiments recorded network traffic from several virtual machines during their normal operation. Figure 7 shows a screenshot of the Wireshark Protocol Hierarchy window. In addition to the number of packets of each protocol in the captured traffic, it also shows which packets were inside which ones, which is not important for the problem under consideration. The figure shows that the number of ICMP

protocol packets (InternetControlMessageProtocol) is very small and amounts to 8 units out of 5000, or 0.2%.

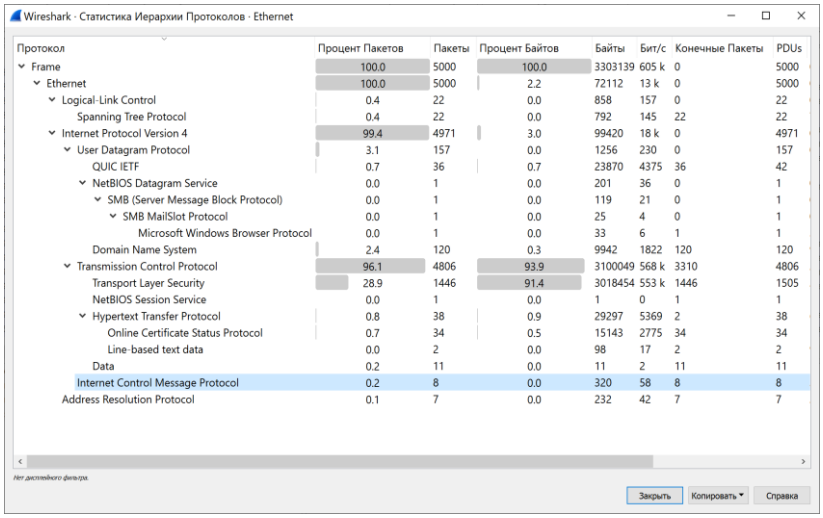

Figure 7: Protocol Statistics for Normal Traffic

The following Figure 8 shows a screenshot of a similar window, but for traffic captured during the exchange of messages hidden inside ICMP packets. The figure shows that the number of such packages has increased to 19 pieces and is 0.4 %.

In general, as can be seen from the presented figures, for any service protocol, the number of its packets relative to the total traffic volumes is always minimal. An increase in the number of service packages indicates potential problems in the network and indicates the need for more thorough verification. To do this, you can conduct a more thorough analysis of the packets of the protocol that caused suspicion, in particular, check the distribution of packet sizes.

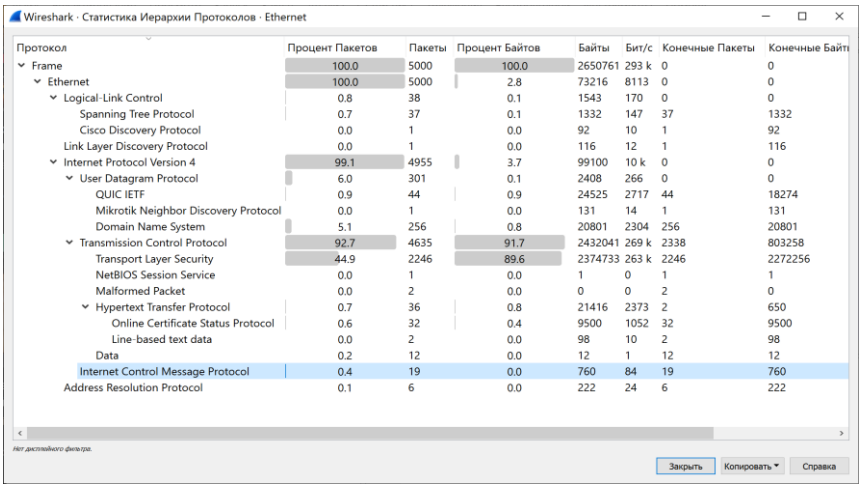

Figure 8: Protocol Statistics for Traffic with Hidden Messages

Unlike data packets, the size of which determines exactly what data needs to be transmitted to the application that generated the packet, service protocols usually have a limited set of transmitted data, such as a variety of requests, response, state codes, etc. Even if the service packet transmits some data that cannot be determined in advance in the future. Since they relate to a specific network, the format of this data is still likely to assume a fixed size, such as addresses and masks. Thus, although service protocol packages may have different sizes, the number of different packet variants will be limited. This is demonstrated in Figure 9, which shows a screenshot of normal Wireshark traffic with filtered ICMP packets. It contains пакеты3 types of packets: Information request, Echo request, and Echo reply. However, all Information request packets are длину60 bytes long, while Echo request and Echo reply are 74 bytes long (Cherckesova et al., 2024).

Figure1-0 shows an example of other captured traffic, where ICMP packets are generated programmatically and contain text messages. Since the length of such messages may vary, the packet size also varies. It differs from the standard size of a normal packet of the type that was chosen for message injection (in this case, it is an Echorequest), and the size of the packets themselves differs, even though they are all formally of the same type.

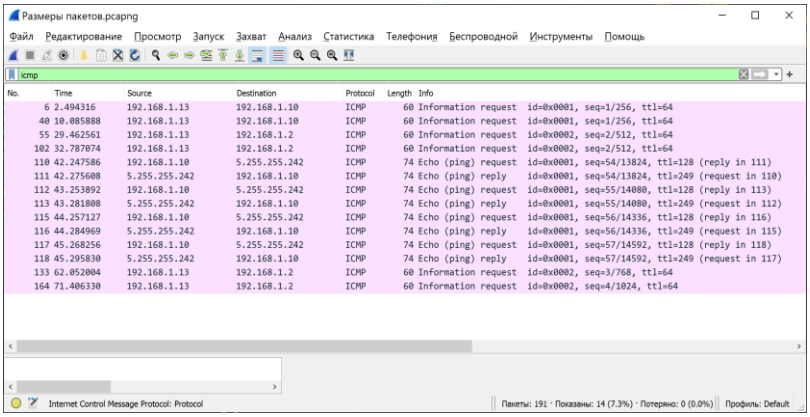

Figure 9: Normal Traffic ICMP Packet Sizes

Such a deviation in traffic characteristics indicates non-standard operation of the ICMP protocol and may indicate the presence of a hidden communication channel.

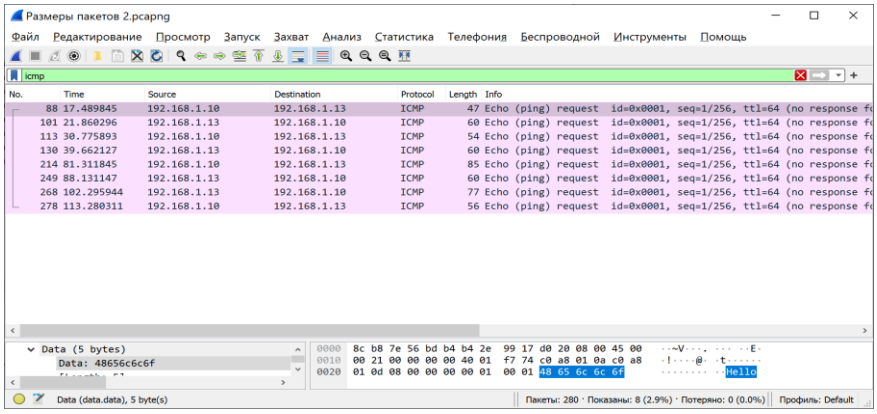

Figure10: Size of ICMP Packets with Embedded Messages

CONCLUSIONS

As a result of the conducted experiments on traffic analysis, it was found that statistical characteristics of traffic can be used to identify steganographic messages that use service network protocols. Thus, the first sign of transmission of hidden (additional) data is a change in the proportions of packets related to service and transport protocols, shown in Figures 8 and 9. In "normal" traffic, the share of service packets is minimal. The use of network steganography tools leads to a significant (in the considered example, 2 times) increase in the share of service packages.

Another important criterion for identifying shorthand data is the size spread of similar packets of the same protocol. As shown in Figure 10, the implementation of messages in service packages using the developed application results in a variation in their size. This parameter distinguishes the received traffic from the one in which steganography was not used. Sharing the presented criteria makes it possible to identify messages embedded in service network packets using the steganography methods described and implemented in the paper.

Thus, the methods and tools considered in the work, depending on the field of use, allow both to detect attempts to use steganographic methods for hidden data transmission, and to create their own secure communication channels, thus increasing the security of telecommunications systems.

REFERENCES

Malyuk AA. Information security: conceptual and methodological foundations of information protection: a textbook for universities. Moscow: Goryachaya liniya-Telekom; 2004.

Shcheglov AY. Protection of computer information from unauthorized access. St. Petersburg: Nauki Tekhnika; 2009. 384 p.

Anin BA. Protection of computer information. St. Petersburg: BHV-Petersburg; 2000. 384 p.

Pavlin DV, Makosii AI, Zhdanov ON. The network steganography. Implementation of the algorithm RSTEG. Reshetnev Readings 2014;18(2):322–324.

Golubev EA, Emelyanov GV. Steganography as one of the directions of ensuring information security. T-Comm – Telecommunications and Transport 2019; S3:185–186.

Cherckesova L, Revyakina E, Buryakova O, Gazizov A. Creation of an encryption algorithm resistant to attacks through side channels of leakage. E3S Web of Conferences 2024; 583:06011. https://doi.org/10.1051/e3sconf/202458306011

Cherckesova L, Revyakina E, Safaryan O, Porksheyan V, Kazaryan M. Analysis of the possibilities of carrying out attacks on the functions of transferring control to operating system console using active intelligence methods. International Research Journal of Multidisciplinary Scope 2024;5(2):516–534. https://doi.org/10.47857/irjms.2024.v05i02.0558

Ganzhur MA, Dzyuba YaV, Panchenko VA. Features of digital steganography as a method of ensuring data hiding. Problemy sovremennogo pedagogicheskogo obrazovaniya 2018;59-4:10–15.

Kodatsky N, Revyakina E, Gazizov A, Gunko V. Using machine learning to forecast hard drive failures. E3S Web of Conferences 2024;549:08024https://doi.org/10.1051/e3sconf/202454908024

Peskova OYu, Halaburda GYu. Application of network steganography for protection of the data transferred over the internet. Izvestiya SFEDU. Engineering Sciences 2017;12(137):167 176.

Sorokin SV. Development of network applications using the TCP/IP protocol stack: textbook. Tver: Tver State University; 2020. 80 p.

Belkina TA. Analytical review of the use of network steganography for solving information security problems. Molodoy uchenyy 2018;11(197):36–44.

Galatenko VA. Standards of information security: a course of lectures. Moscow: Internet University of Information Technologies; 2006. 264 p.

Frączek W, Szczypiorski K. Perfect undetectability of network steganography. Security and Communication Networks 2016;9:2998–3010. https://doi.org/10.1002/sec.1491