ISSN 0146-4116, Automatic Control and Computer Sciences, 2016, Vol. 50, No. 8, pp. 693–702. © Allerton Press, Inc., 2016.
Original Russian Text © V.V. Nikonov, V.P. Los', G.V. Ross, 2016, published in Problemy Informatsionnoi Bezopasnosti, Komp'yuternye Sistemy.

Development of Automated System for Identifying Abnormal Network Activity and Detecting Threats

V. V. Nikonov, V. P. Los'*, and G. V. Ross

Moscow Technical University (MIREA), Moscow, 119454 Russia

**e-mail: los-vladimir@yandex.ru*

Received June 16, 2016

Abstract—The features of a system for identifying abnormal network activity are considered. Algorithmic and software systems for identifying abnormal network activity and detecting threats are developed.

Keywords: abnormal network activity, network threat, agents' log, activity profile, node's network card, socket, winsock library

DOI: 10.3103/S0146411616080150

FEATURES OF DATA ANALYSIS AND REPRESENTATION SYSTEM IMPLEMENTATION

System agents collect data on local stations and provide protection for the network and its components and keep the activity log. But for effective network protection, a specialist who knows how to analyze the data and take measures to counter the attack is needed. For networks consisting of a small number of workstations it is quite easy to monitor the state from the database based on the activity log. But if the protected network has hundreds of workstations, then it is impossible to track all threats without special tools. It is proposed to use data analysis and the representation system as such tools.

It is advisable to divide this system into two subsystems*: the system of agents' activity log analysis and the data representation system*.

Features of the Agents' Log Analysis System

The system analyzes the log for distributed attacks in time and for detection of potentially dangerous network sites.

Agents on the workstations are not able to track attacks that are distributed in time because they are handled in the stream of data, and many of the attacks, as far as any particular data packet is concerned, are irrelevant or even normal activity. For example, sorting passwords distributed in time will not generate the suspicions of the agent [1]. And in the case of log analysis, the system identifies attempts of an incorrect password entry and alerts the system administrator. Features of system implementation are shown in Fig. 1.

The system obtains data from the database and analyzes it for network threats. The analysis can be performed for a certain period of time, both daily and hourly. The possibility of false positives is high in the case of detecting distributed attacks, so the system only reports on its suspicions to the system administrator, who must take appropriate measures to eliminate the threat.

Analyzing the activity of agents based on statistical data, the system identifies the nodes in the network, which are most often subjected to threat, and provides data on these nodes to the system administrator, which allows taking additional measures to protect the vulnerable segment.

Features of the Data Representation System

The system sorts the data from the activity log and provides that data to the system administrator through a flexible interface.

The data representation system is an essential tool in work with large volumes of information.

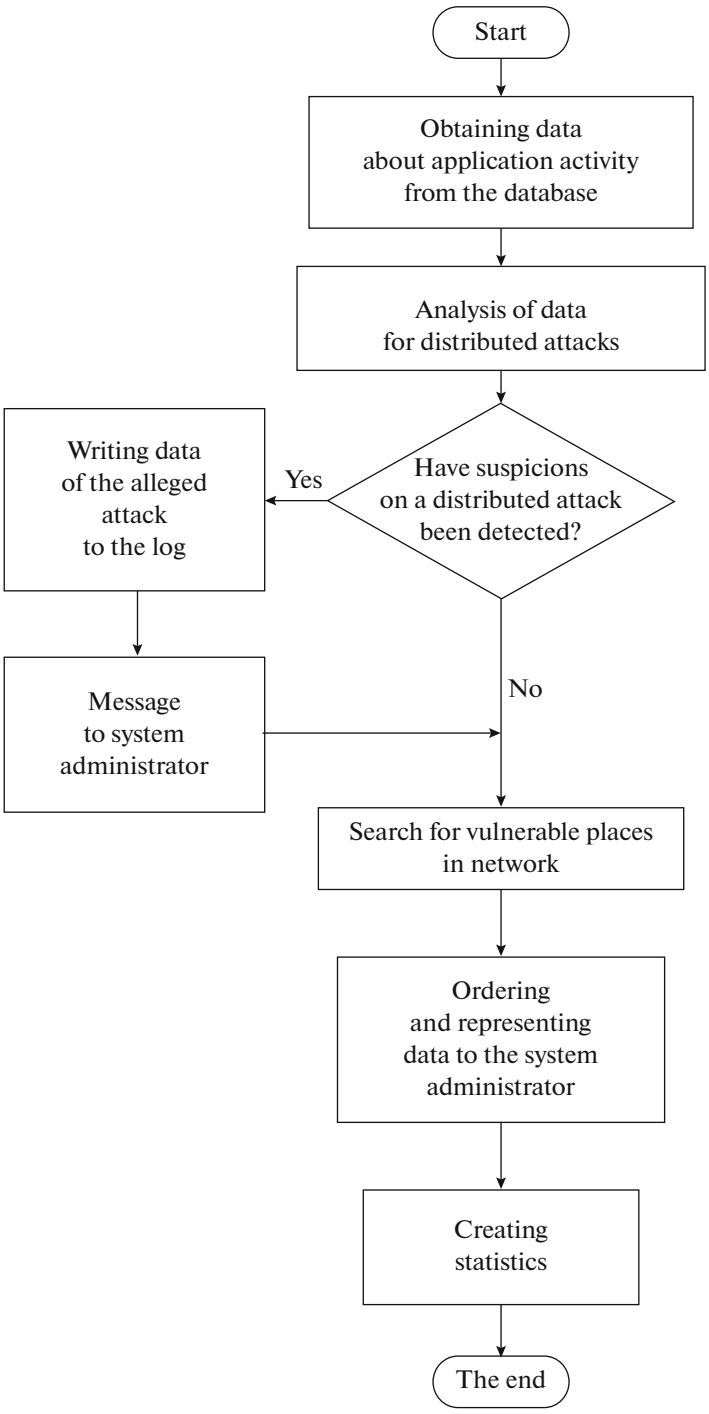

Fig. 1. Implementation of data analysis system.

The system receives data from the database, sorts the data by identifying the highest priority, and separating it on the network segments. It generates statistics about the number of prevented threats, user activity and traffic over a certain period of time.

It provides the data to system administrator, who are responsible for their own network segments, in a form easy to view and analyze.

The interface through which the system administrator views the data should be flexibly adapted to the user. Necessary functions are sorting of data by host, by threat type, by time, and so on. The statistics are

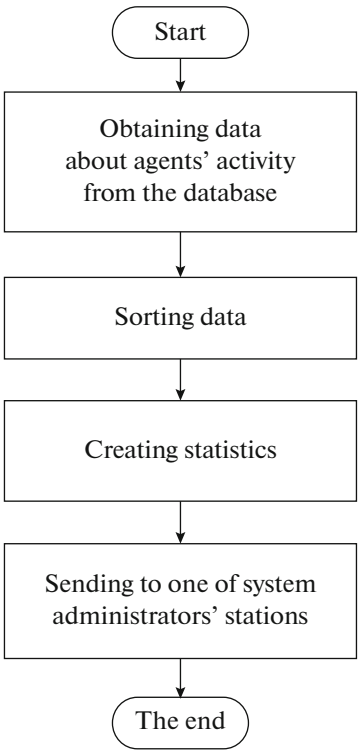

Fig. 2. Implementation of data representation system.

presented by graphs and charts, in user-friendly form. The interface also allows real-time tracking processes and services that run on the personal stations, and characteristics of the network and user activity.

ALGORITHMS FOR IDENTIFYING ABNORMAL NETWORK ACTIVITY

Construction of Activity Profiles

Some agents in their work assume construction of activity profiles: for anomalous traffic, application activity, and user authentication.

For creating profiles, it is proposed to use correlation analysis, because this method is low-cost for the workstation resources, which allows its effective use without disturbing the user. The analysis for anomalous activity occurs by comparing the normal values of the autocorrelation function with the current values.

Activity indices are read on workstations constantly, but quantitative values for the designated period of time are needed to create the autocorrelation function. They are obtained by subtracting the previous values from current ones after a specified time.

In order to collect information about network activity it is proposed to use the WinSock2 library at this stage of development. The library allows scanning input packets at the network card of the computer and performing their subsequent analysis. Windows Sockets 2 is a handy tool for creating a network traffic analyzer for Windows. In comparison with other solutions, it is easier and less costly for the resource workstation. It is proposed to use the Pcap library to provide cross-platform use of the agent in its further development. Pcap is a more powerful library and it can be implemented under Windows, Linux, and Macintosh.

It is proposed to use the User32 library to collect information about the user's activity by polling keystrokes and the mouse by searching for keywords in the resulting data [2].

The network activity profile is a set of autocorrelation functions of the following parameters with respect to time: average length of data packets, duration of peak traffic, duration of session, number of sent packets, number of received packets, average lifetime, and number of lost packets.

Also, the profile of applications consists of a set of autocorrelation functions built on the following parameters: the CPU load, RAM load, load of information carrier, and the communication channel load.

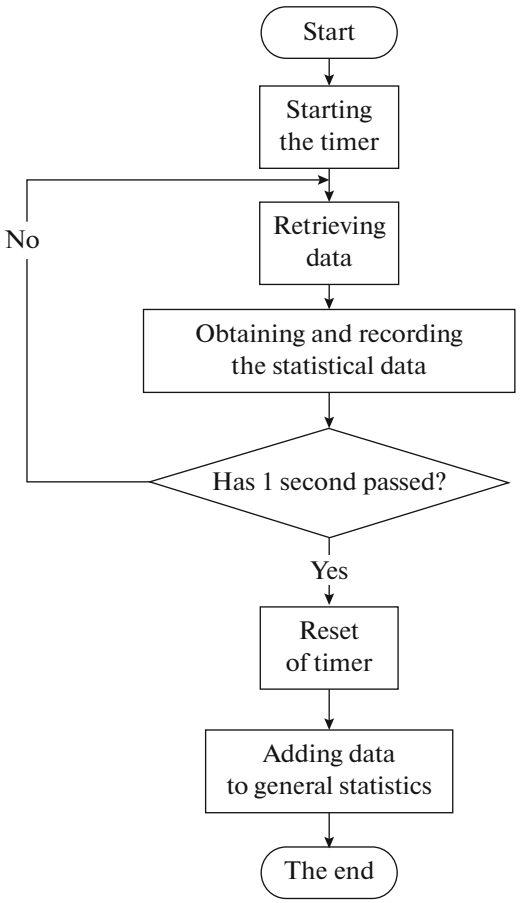

Fig. 3. General algorithm for obtaining data for the autocorrelation function.

A user's profile is based on other indicators, viz., a set of autocorrelation functions, and statistical analysis including the frequency of requests during the session, intervals between requests, speed of mouse movement, typing speed on the keyboard, visiting of directories and editing files, and opening pages in a browser.

An algorithm for constructing profiles is shown in Fig. 3.

Retrieved data and algorithms used are different for the construction of profiles by different agents.

It is necessary to obtain from a received data packet the following information: the IP of sender and receiver, the packet's volume and lifetime, the number of sent and received packets per second, average lifetime, average length of packet, and number of bytes transmitted per second.

The algorithm for obtaining data to create the traffic profile is shown in Fig. 4.

For the agent of application activity, it is sufficient to obtain statistical data about applications running in the system. It is necessary to obtain and calculate the speed of the mouse movement and the number of characters printed per minute for the user's authentication agent. The speed of mouse movement is calculated by segments traversed per unit of time by the mouse cursor on the screen. For this purpose, mouse coordinates x and y are recorded each unit of time. And current coordinates' values are subtracted from the last ones:

$$X = X_2 - X_1,$$

$$Y = Y_2 - Y_1.$$

Obtained values of X and Y are distance that mouse have passed along the abscissa and ordinate axes. Then, the length of the segment in which the mouse was moved directly is calculated using the Pythagorean Theorem.

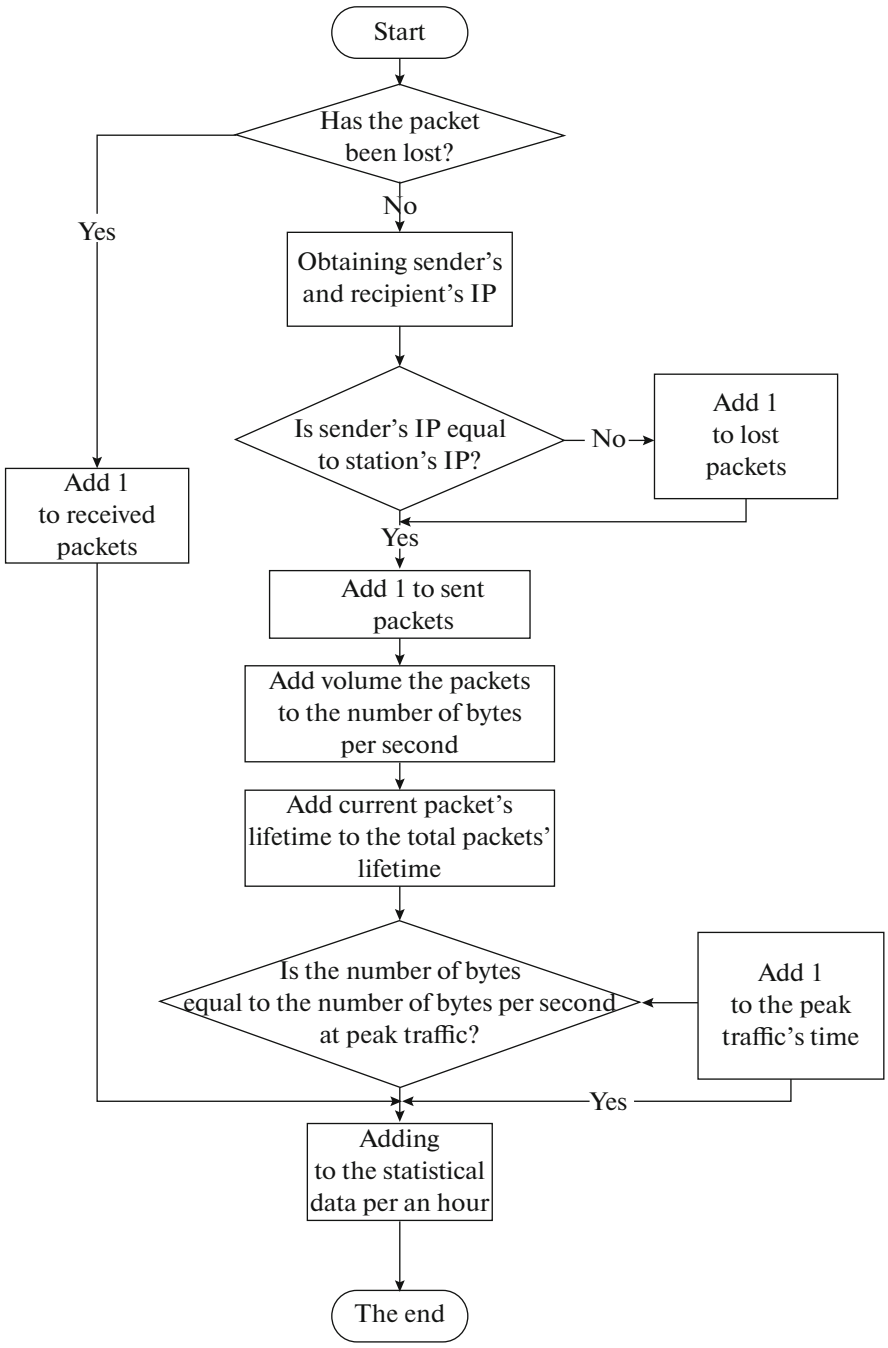

Fig. 4. Algorithm for obtaining statistical information from data packets.

Data about the activity of applications and from the input devices are shown in Figs. 5 and 6, respectively.

Current figures of autocorrelation functions are compared with profile figures for a certain time portion in order to identify abnormal activity.

Detection of abnormal network activity is based on the construction of various types of profiles. As described in the special section, all the profiles are constructed mainly based on the autocorrelation functions. Receiving data, on which necessary features could be built, is one of the most important tasks for the development of this system. Obtaining data using the least resources of workstations is the priority task in the case of a multi-agent system.

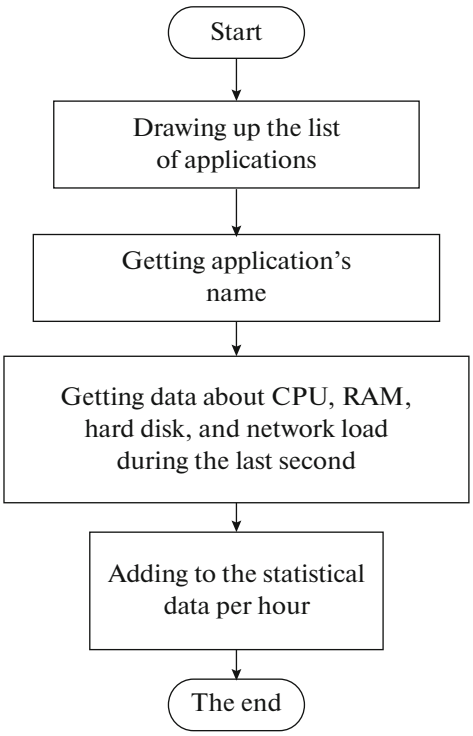

Fig. 5. Algorithm for obtaining data about applications' activity.

RECEIVING THE DATA

The WinSock library is used to obtain data about traffic passing through a specific node [3]. Sockets are a high-level unified interface for interaction with telecommunication protocols [4].

The essence of this method is to create a "raw" socket, which allows direct access to packets that are forwarded via the network.

To create a "raw" socket:

```
s = socket( AF_INET, SOCK_RAW, IPPROTO_IP );
```

It is necessary to switch socket into a listening mode so that it can receive all the packets passing through node's network card:

```
// Flag PROMISC On/off
unsigned long flag = 1;
SOCKET s; // Listening socket.
//Constant from Mstcpip.h library
#define SIO_RCVALL 0x98000001
// Enable promiscuous mode.
ioctlsocket(s, SIO_RCVALL, &flag);
```

Then infinite loop for reading and parsing packets into components starts.

```
IPHeader* hdr = (IPHeader *)Buffer;
sa1.s_addr = hdr->iph_src; //Sender's Ip
sa1.s_addr = hdr->iph_dest;//Recipient's Ip
// Calculating the volume. Since the direct order instead of the reverse
// is used in the network, then it is necessary to swap bytes
lowbyte = hdr->iph_length>>8;
hibyte = hdr->iph_length<<8;
hibyte = hibyte + lowbyte; //Packet's length
ttl=itoa(hdr->iph_ttl,temp_buffer,10); //Packet's lifetime
```

Obtaining data about computer's user action is carried out using User32.dll library. The essence of the process for obtaining the data entered from the keyboard is reduced to the interception of functions:

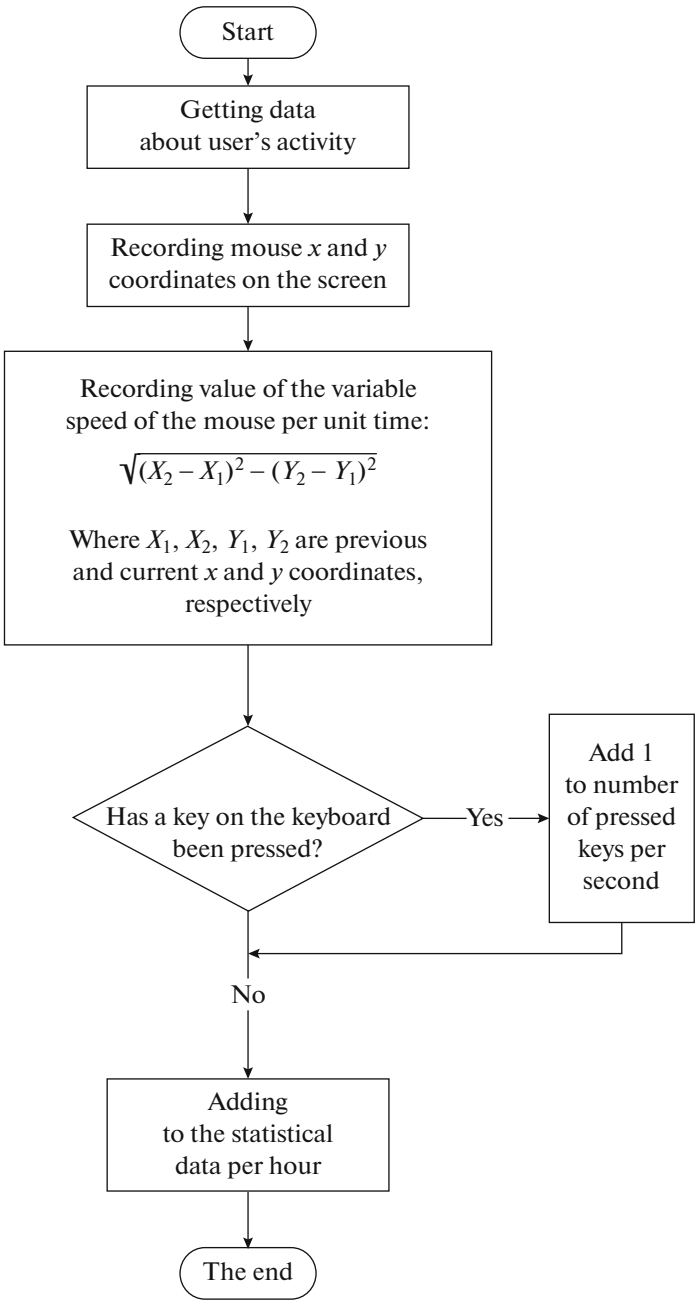

Fig. 6. Algorithm for obtaining data from input devices.

```
GetMessage(&msg, hwnd, 0, 0);
```

or

```
PeekMessage(&msg,0,0,0,PM_REMOVE);
```

These functions allow one to control the information entered from the keyboard [5, 6]. The technique consists of the interception of functions such as PeekMessage caused by the application and call its own PeekMessage from User32.dll, similar to the application. If the function returns true, then it means that the message has been received and stored in the exchange buffer &msg:

```
while (PeekMessage(&msg,0,0,0,PM_REMOVE)) {
// Necessary actions
}
Sleep(10);
```

Next stage is checking the buffer (&msg) for messages: WM_KEYDOWN, WM_KEYUP, WM_CHAR. The pressed key's code could be easily found out and logged after receiving these messages. This method requires frequent monitoring the keyboard, in this case, every 0.01 seconds.

BUILDING THE AUTOCORRELATION FUNCTION

The autocorrelation functions are based on statistical data obtained by agents. This information is kept in structures that are easy to both store and transmit over the network, and to extract the necessary parameters. Features of structure's composition can be considered using an example of the traffic profile:

```
struct traffic_profile
{
    double bps; //bytes per second
    double getpps; //packets received per second
    double sendpps; //packets sent per second
    double pack_size; //average length of the packet
    double pack_ttl; // average packet's lifetime
    double i; //number of requests
    //current figures
    double bps_now;
    double getpps_now;
    double sendpps_now;
    double pack_size_now;
    double pack_ttl_now;
    int i_now;
};
traffic_profile hour[24].
```

Data that was obtained by the agent from the received packets are added to the current figures and statistics for the count per hour is performed. Data from the current figures are added to the general at the end of the hour. There is a need to increase the "weight" of new indicators relative to those previously collected for improvement of the function adjustment process.

$$bps = \frac{(bps \cdot i) + \left(bps\_now \cdot \frac{(i+1)}{10}\right)}{i + \frac{(i+1)}{10}}, \quad (1)$$

where:

bps is the total number of bytes per second sent over the network;

bps_now is the current number of bytes per second sent over the network;

i is the number of measurements.

It is seen from the formula (1) that the new incoming information (*bps_now*) is added to the old is always a tenth part of that which makes it much faster to adjust the traffic, given varying indicators. This allows us to quickly adjust the autocorrelation function in the case of new tasks and responsibilities of employees arising, and to reduce the number of false positives. Functions for other indicators are built in similar way:

$$getpps = \frac{(getpps \cdot i) + \left(getpps\_now \cdot \frac{(i+1)}{10}\right)}{i + \frac{(i+1)}{10}}, \quad (2)$$

where:

getpps is the total number of received by the station packets;

getpps_now is the current number of received packets.

$$sendpps = \frac{(sendpps * i) + \left(sendpps\_now * \frac{(i+1)}{10}\right)}{i + \frac{(i+1)}{10}}, \quad (3)$$

where:

sendpps is the total number of sent packets;

sendpps_now is the current number of sent packets.

$$pack\_size = \frac{(pack\_size * i) + \left(pack\_size\_now * \frac{(i+1)}{10}\right)}{i + \frac{(i+1)}{10}}, \quad (4)$$

where:

pack_size is the average volume of a single data packet;

pack_size_now is the current value of the average volume of a single data packet.

$$pack\_ttl = \frac{(pack\_ttl * i) + \left(pack\_ttl\_now * \frac{(i+1)}{10}\right)}{i + \frac{(i+1)}{10}}, \quad (5)$$

where:

pack_ttl is the average packet's lifetime;

pack_ttl_now is the current value of the average packet's lifetime.

The current indicators (**_now*) are formed basing on the conventional statistical data with equal weight, regardless of the time of receipt.

$$bps\_now = \frac{bps\_now * i\_now + localsize}{i\_now + 1}, \quad (6)$$

where:

localsize is the sum of all the bytes of packets collected per second.

$$getpps\_now = \frac{getpps\_now * i\_now + localgetpps}{i\_now + 1}, \quad (7)$$

where:

localgetpps is the number of received packets per second.

$$sendpps\_now = \frac{sendpps\_now * i\_now + localsendpps}{i\_now + 1}, \quad (8)$$

where:

localsendpps is the number of sent packets per second.

$$pack\_size\_now = \frac{pack\_size\_now * i\_now + \frac{localsize}{localsendpps + localgetpps}}{i\_now + 1}, \quad (9)$$

where:

localgetpps is the number of received packets per second.

$$pack\_ttl\_now = \frac{pack\_ttl\_now * i\_now + \frac{localttl}{localsendpps + localgetpps}}{i\_now + 1}, \quad (10)$$

where:

localttl is the sum of the lifetime of all packets collected per second.

Common and current values for each period of time can be seen in numerical form, and on the graph (Fig. 7).

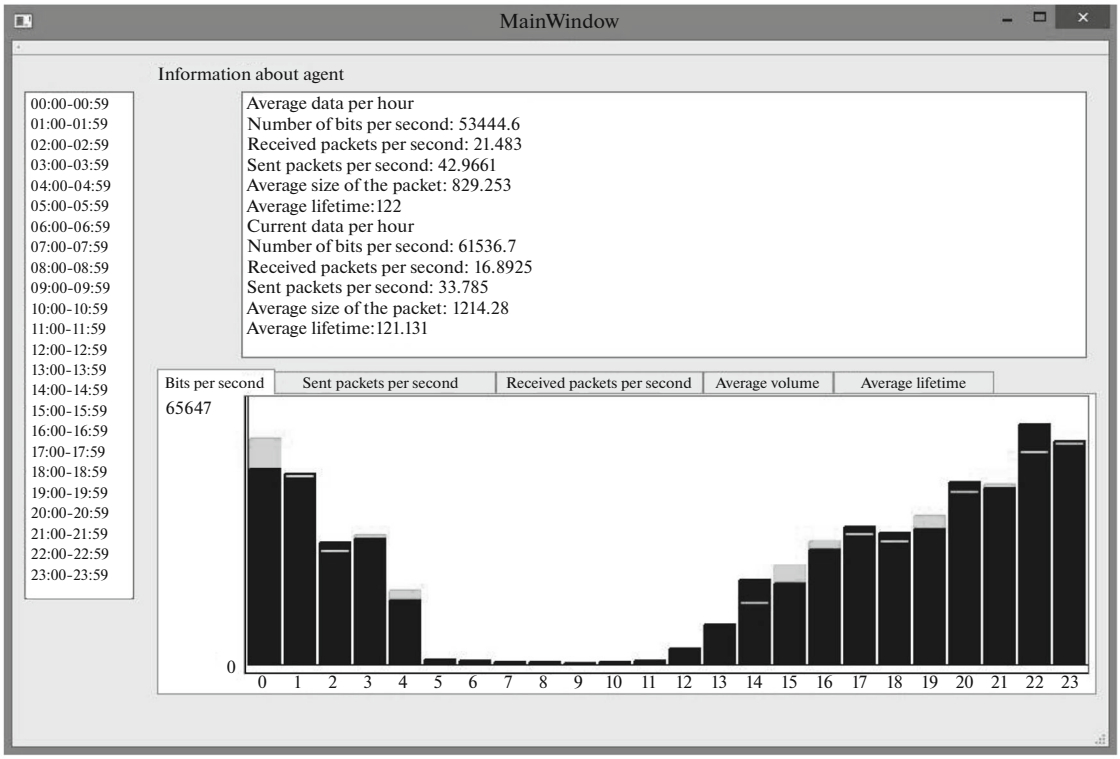

Fig. 7. Presentation of the quantitative characteristics of traffic in numerical from and in the form of the autocorrelation function.

For storage of quantitative profiles, agents send the information to the database. Then, it is forwarded to the data representation system.

CONCLUSIONS

A multiagent system for identifying abnormal network activity and detecting threats has been developed.

The algorithms of the agents, log analysis and data representation subsystems have been developed. The general strategy for the search for abnormal activity using constructed profiles (traffic, applications, and users) is determined in this system. It works using the method of autocorrelation functions.

The mathematical basis of autocorrelation function composition was developed.

Test versions of the network activity analysis agent and data representation systems were developed.

REFERENCES

1. Peleshenko, V.S., Development of a mathematical model of information exchange in the local area network for the realization of assets and the method of network security information, *Cand. Sci. (Eng.) Dissertation,* Stavropol, 2007.
2. Zavgorodnii, V.I., *Kompleksnaya zashchita informatsii v komp'yuternykh sistemakh* (Comprehensive Protection of Information in Computer Systems), Moscow: Logos, 2001.
3. Soshnikov, I.O., Frolov, U.A., Plyukhina, I.A., and Peleshenko, V.S., Threats to information security in computer networks and troubleshooting, *Vopr. Nauki,* no. 4, (11), 2014, pp. 20–26.
4. Shan'gin, V.F., *Zashchita informatsii v komp'yuternykh sistemakh i setyakh* (Protection of Information in Computer Systems and Networks), Moscow: DMK Press, 2012.
5. Yarochkin, V.I., *Informatsionnaya bezopasnost': Uchebnik dlya studentov vuzov* (Information Security: Textbook for University Students), Moscow: Academic Project; Gaudeamus, 2004, 2nd ed.
6. Belov, E.B., Los', V.P., Meshcheryakov, R.V., and Shelupanov, A.A., *Osnovy informatsionnoi bezopasnosti* (Fundamentals of Information Security), Moscow: Goryachaya liniya, 2006.

Translated by A.V. Viazovtsev