

AI and LLM Models to Analyze and Identify Cybersecurity Incidents

Miroslava Ruzickova ¹, Irada Dzhalladova ², Oleg Kaminsky ², Oleksandr Bartash ² and Arsen Pavlov ²

¹ University of Białystok, Faculty of Mathematics and Informatics, Białystok, Poland

² Kyiv National University of Economics named after Vadym Hetman, 54/1 Prospect Peremogy, 03057 Kyiv, Ukraine

Abstract

Many applied methods of cyber security require time-consuming calculations, which requires the use of specialized software for their implementation. Therefore, the issue of using artificial intelligence tools in cyber security analytics to automate routine tasks remains relevant.

The research examines the analysis of firewall logs to detect cybersecurity incidents using large language models, artificial intelligence, Python, and the Pandas library in order to abstract the cybersecurity analyst from writing software code and allow him to focus only on creating the right tasks for AI systems. Also, the article proposes a model for evaluating the effectiveness of technical information protection against unauthorized access using a functional approach.

Keywords ¹

Cybersecurity AI, Long Language Model, ChatGPT, random process

1. Introduction

We live in the information age, where there has never been such an abundance of information sources in history. The field of cybersecurity is no exception. Among the most valuable data sources for analyzing cybersecurity incidents are server logs and firewalls. These logs provide information about network connections and internal organizational traffic, and in some cases, even user access to VPNs. In this era of heightened threats, the integration of artificial intelligence (AI) and large language models (LLMs) becomes a transformative force in the field of cybersecurity.

A large language model is a type of generative artificial intelligence language model that stands out for its ability to achieve general understanding and generate language. Essentially, it's an algorithm that feeds on a "large" or massive dataset to learn the relevant syntax of language. Thanks to its understanding, a large language model can interpret, analyze, and generate synthetic human-like sentences or textual information. Notable examples include OpenAI's GPT models (such as GPT-3.5 and GPT-4, DALL-E, used in ChatGPT), Google's PaLM (used in Bard), and Meta's LLaMa [1].

In a study [2], the development process of programs using GPT-4 and ChatGPT was analyzed. Clear and detailed explanations of artificial intelligence concepts were provided, along with practical guidelines for effective, secure, and economical integration of OpenAI services.

Analysis of cybersecurity incidents involves a deep examination that determines the level of danger, extent of damage, and losses, as well as the detection of artifacts (traces or samples of malicious software). In the work by E. Chou [3], the application of high-level Python packages and frameworks is discussed for tasks related to network automation, programming, and security data analysis, including Azure and AWS Cloud. For those who wish to become more deeply acquainted with the object-oriented language Python, we suggest reading one of the most famous books [4].

Dynamical System Modeling and Stability Investigation (DSMSI-2023), December 19-21, 2023, Kyiv, Ukraine

EMAIL: m.ruzickova@math.uwb.edu.pl (A.1); idzhalladova@gmail.com (A.2); olkam@kneu.edu.ua (A.3); petrgabriluch@gmail.com (A.4); pavlov_arsen@outlook.com (A.5)

ORCID: 0000-0002-7724-763X (A.1); 0000-0003-3158-6844 (A.2); 0000-0003-0607-8944 (A.3); 0009-0001-2283-6402 (A.4); 0009-0003-9252-7415 (A.5)



© 2023 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

The study [5] explores the interplay between artificial intelligence, machine learning, and deep learning, analyzing the impact of large language models such as ChatGPT and Bard [6] on society and professional competencies.

Additional difficulties also arise when complex models are described most adequately, for example, taking into account the time deviation of the argument [7,8].

Many applied cybersecurity methods require labor-intensive calculations, necessitating the use of specialized software for their implementation. Therefore, the question of applying artificial intelligence tools in cybersecurity analytics to automate routine tasks remains relevant.

2. Main results

Problem Statement: According to Gartner, Inc [9], security department experts must reassess their investment balance between protective technologies and a human-centric approach to cybersecurity when developing and implementing enterprise cybersecurity systems, in line with new technological trends.

Let's first define the problem we need to address: conducting an analysis of firewall logs in the security system to identify data artifacts.

A log is a text file containing information about software actions or user activities, stored on a computer or server. It serves as a chronological record of events and their sources, errors, and reasons behind them.

Log analysis is a fundamental tool for cybersecurity professionals. It helps uncover the sources of various issues, detect conflicts in configuration files, and track security-related events. However, reading and analyzing logs is only possible with specialized software.

Let's consider a model for evaluating the effectiveness of technical information security against unauthorized access using a functional approach. The essence of the functional approach is as follows:

Let there be an information system (IS) where, according to regulatory requirements, a certain set of protective measures F_k must be applied to achieve a specified level (class) of security up to $k = \overline{1, K}$. However, in practice, only a subset of these protective measures, denoted as f_k , has been implemented in the IS from the set F_k . All combinations of protective measures from the F_k set can be arranged in ascending order of their effectiveness, i.e., their impact on enhancing information security in the IS.

Yes, according to [10], if an information system (IS) needs to be protected at the third level of security, it should implement 63 protective measures ($F_k = 63$). The number of combinations of such protective measures, denoted as N_k , would be equal to $2^{F_k} - 1 = 2^{63} - 1$. As we transition from one combination to another, the effectiveness of protection increases. An approximate indicator of information security effectiveness can be expressed by the following ratio:

$$\eta_k = \frac{1}{N_k} \sum_{n_k=1}^{N_k} \prod_{f_k \in n_k} \delta(f_k), \quad (1)$$

where $\delta(f_k)$ - is the Kronecker delta function, which equals 1 if the protection measure with the number f_k , included in the combination n_k , is implemented in the system, and 0 otherwise.

Illustration of the dependence of the effectiveness of third-level security protection on the current combination number of protective measures n_k is shown in Figure 1.

Instead of formula (1), an approximate assessment can be expressed using the following ratio:

$$\eta_k = \frac{2^{n_k} - 1}{2^{N_k} - 1},$$

where n_k^* represents the number of combinations of events in which all the events included in the combination occur. For a sufficiently large number of combinations, n_k can be approximated using the following formula:

$$\eta(V_k) \approx \frac{1}{2^{N_k(1-v_k)}},$$

where $v_k = n_k^* / N_k$ represents the proportion of realized protection measures from the total number of protection measures that need to be implemented in the system.

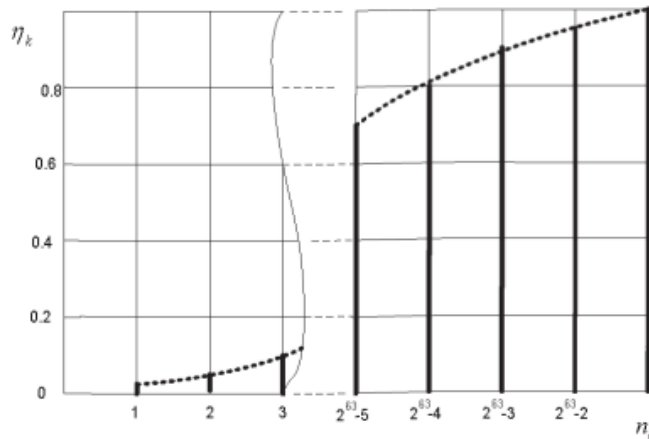


Figure 1. Illustration of the dependence of information security protection effectiveness on the combination of protective measures for the third level of IS security.

The relationship between the effectiveness of protection and the proportion of implemented protective measures in a third-class security system is shown in Figure 2.

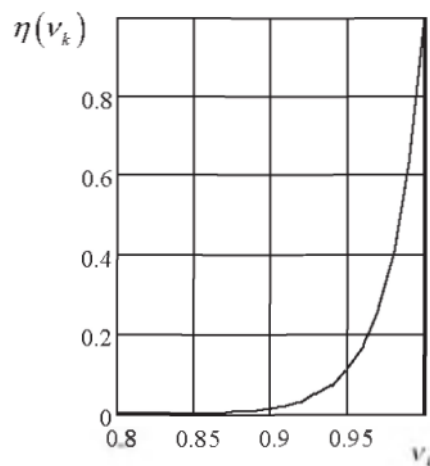


Figure 2. The relationship between protection effectiveness and the proportion of implemented information security measures in third-class security systems.

Using a human-oriented approach, it is possible to propose the following hypothesis:

Hypothesis 1: Abstract cybersecurity analysts from writing code and allow them to focus solely on formulating the right questions for artificial intelligence systems.

The automation of tasks is the primary purpose of systems based on artificial intelligence. Language models have always been able to perform syntactic analysis, identify patterns in datasets and texts. On the other hand, large language models have advantages in semantic analysis, allowing them to understand basic meanings and context, thereby achieving higher accuracy.

The OpenAI team has developed a highly intuitive Python SDK that can be easily installed using the pip package manager. To get started, you can install it with the following command:

```
pip install openai
```

To use the OpenAI API, you'll need an API key. You can register for an API key on the OpenAI website <https://openai.com> and create a key in the API keys section.

```
import openai  
openai.api_key = "Your API Key"
```

Replace the value of the parameter “Your Key” with the API key obtained from the OpenAI platform page. Now it is possible to prompt the user using the `input ()` function:

```
question = input ("What would you like to ask ChatGPT? ")
```

The `input ()` function is used to prompt the user to enter a question they would like to ask the ChatGPT API. The function takes a string as an argument, which is displayed to the user when the program is run.

To pass the user’s question from your Python script to ChatGPT, you will need to use the ChatGPT API completion function.:

```
from openai import OpenAI
client = OpenAI()

response = client.completions.create(
    model="gpt-3.5-turbo-instruct",
    prompt=" What cyber security incidents do you know?"
)
```

The `client.completions.create()` function in your code is used to send a request to the ChatGPT API for generating completions based on the user’s input prompt. The `model` parameter allows you to specify a specific variant or version of the GPT model you’d like to use for processing the request, and in this case, it’s set to “gpt-3.5-turbo”. The `prompt` parameter defines the textual prompt for the API execution, which in this scenario is the user’s question.

By passing contextual information and questions to the function in text format, the responses will also be obtained in text format. It’s essential to recognize that while ChatGPT performs well in answering general questions and providing solutions for moderately complex problems, it’s not infallible. In cases where complex problem-solving requires expert reasoning and context understanding, artificial intelligence needs sufficient information to provide accurate tools for the task at hand [11].

Hypothesis 2: ChatGPT requires context and more information to understand a problem fully. In some cases, it needs to go through a logical process to achieve the desired goal.

Let’s demonstrate an example response from ChatGPT to a model problem to validate this hypothesis (see Figure 3):

Model Problem 1. *"How do I know the IP address of the command-and-control console?"*

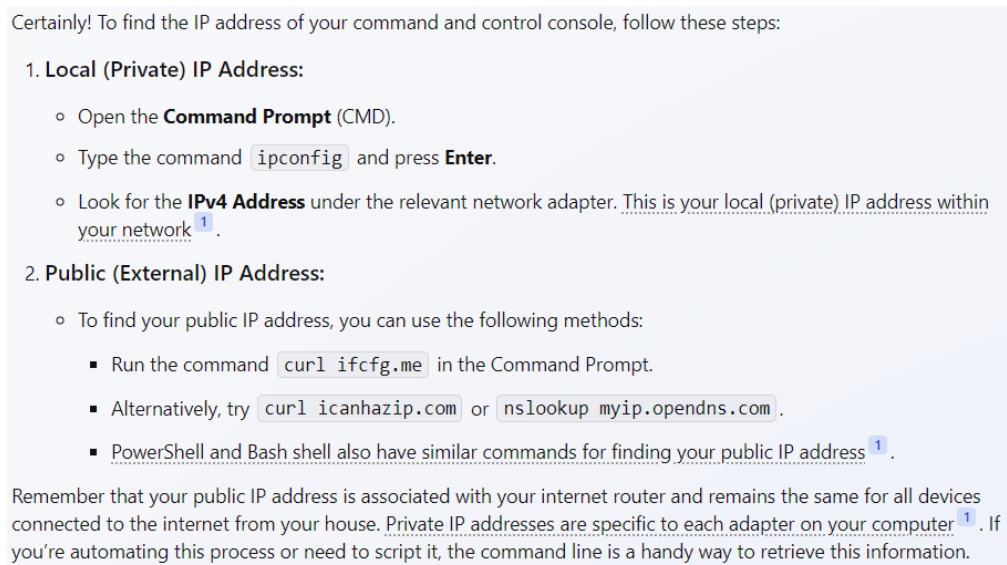


Figure 3: Answer to the request

Understood that this response is not useful for cybersecurity analysis. Let’s analyze to identify what’s wrong with the query structure and discover key factors for improvement:

- Lack of sufficient context in the query.
- Absence of necessary information.
- Missing a clear goal expected from artificial intelligence.
- Lack of specific instructions to achieve the query's purpose.

People often forget the intricacies of the thinking process and assume that artificial intelligence should understand the expert's thought process, rather than the other way around, which is logical. However, when working with LLM, context is a crucial component for enhancing the results obtained. Adhering to this principle, let's supplement your question with context, and I'll strive to provide a more accurate answer.

Model Problem 2. *"Context: The expected outcome is a Python code that will assist the cybersecurity analyst during the investigation of ransomware attacks on the enterprise network. We intend to analyze a file containing data related to the production firewall traffic of the Palo Alto company. The content is already in Pandas DataFrame format, stored in a variable named 'data'."*

The response is shown in Figure 4. As you can see, the result is taking shape, but it is still far from the intended goal. While one of the reasons for using artificial intelligence technology is to obtain information, it is not the primary objective. The more information is provided in queries, the more significant improvements are observed in the responses.

Clearly, in a situation where there is a large volume of logs (for example, a 200 MB \$MFT file), it is unrealistic to send this information to artificial intelligence for processing. Aside from data protection issues, the costs of using the artificial intelligence system's API would be significantly higher than planned.

```
import pandas as pd

# Assuming you have a DataFrame named 'data' with the required columns

# Step 1: Filter traffic to identify potential ransomware activity
ransomware_traffic = data[data['protocol'].isin(['TCP', 'UDP']) & (data['bytes_sent'] > 0)]

# Step 2: Group traffic by source IP to analyze behavior
source_ip_grouped = ransomware_traffic.groupby('source_ip')

# Step 3: Analyze behavior per source IP
for source_ip, group_data in source_ip_grouped:
    print(f"\nAnalyzing traffic for source IP: {source_ip}")

    # Check for unusual protocols or high traffic volume
    unusual_protocols = group_data['protocol'].nunique() > 1
    high_traffic_volume = group_data['bytes_sent'].sum() > 1000000 # Adjust threshold as needed

    if unusual_protocols or high_traffic_volume:
        print("Potential ransomware activity detected:")
        print(group_data)

    # Additional analysis or actions can be performed here

# Step 4: Identify unusual destination IPs
unusual_destination_ips = data.groupby('destination_ip')['bytes_sent'].sum().sort_values(ascending=False)

# Display the top 5 unusual destination IPs
print("\nTop 5 Unusual Destination IPs:")
print(unusual_destination_ips.head())

# Step 5: Check for multiple connections to the same IP in a short time
data['timestamp'] = pd.to_datetime(data['timestamp'])
data['time_diff'] = data.groupby('source_ip')['timestamp'].diff()

# Set a threshold for short time differences, indicating potential rapid connections
short_time_threshold = pd.Timedelta(minutes=5)
rapid_connections = data[data['time_diff'] < short_time_threshold]

# Display potential rapid connections
print("\nPotential Rapid Connections:")
print(rapid_connections[['timestamp', 'source_ip', 'destination_ip', 'time_diff']])
```

Figure 4: ChatGPT software code.

Instead, let's tell the artificial intelligence in the request what the data for analysis looks like, providing it with as much contextual information as possible so that the data itself is irrelevant (see Fig. 5). In this case, we will add to the ChatGPT request a description of the columns that contain logs and a description of each field.

```
Domain: Firewall domain name.
Receive Time: Time when log was received.
Serial #: Firewall serial number.
Type: Type of log (traffic, threat, etc.).
Threat/Content Type: Type of threat or content.
Config Version: Firewall configuration version.
Generate Time: Time when log was generated.
Source address: Source IP address.
Destination address: Destination IP address.
```

Figure 5: Log description

Model Problem 3. *"Context: The expected output is Python code that will assist cybersecurity analysts in their investigation of a ransomware attack on an enterprise network. We are going to analyze a file containing data about the traffic of the Palo Alto company's firewall. The content is already in Pandas DataFrame format in a variable called "data". Columns in this DataFrame: {fields}*

And private ranges of IP addresses:

```
- 10.0.0.0 i 10.255.255.255
- 172.16.0.0 i 172.31.255.255
- 192.168.0.0 i 192.168.255.255"
```

The role of the analyst in such a process is important. What is needed is not just an understanding of how the technology works, but an understanding of the problems and the ability to ask questions, even when the analyst has abstracted from the intermediate process of analysis.

Let's change the general question used earlier to a more specific prompt that can guide the analysis process. Let's use the same context as before, but change the question (answer in Fig. 6):

Model Problem 4. *List the external IP addresses to which the most connections were made when browsing the web between 6:00 PM and 8:00 AM, and show me in a column the number of unique source IP addresses that connected to each of the external IP addresses address.*

```
# Assuming you have a DataFrame named 'data' containing relevant traffic data
# Filter data for the specified time range (18:00 to 8:00)
filtered_data = data[(data['Timestamp'] >= '18:00') | (data['Timestamp'] <= '08:00')]

# Group by external IP address and count unique source IP addresses
connection_counts = filtered_data.groupby('External_IP')['Source_IP'].nunique()

# Sort by connection counts in descending order
sorted_connections = connection_counts.sort_values(ascending=False)

# Display the top external IP addresses and their corresponding unique source IP counts
print(sorted_connections.head())
```

Figure 6: A response to a request that has been modified

By using this code, only one IP address will be obtained, and this is a rather suspicious situation. Investigating incidents of this type requires detailed analysis of the fact that only two source IP addresses are making HTTP connections during off-hours.

It is necessary to indicate to the artificial intelligence the correct way to achieve the desired result. In some cases, analysts clearly understand the path to follow to achieve the right goal, but it is not necessary to spend time searching for Python functions that will lead to the result, i.e. the analyst

knows what is needed, but not how to do it, and this is why use chatGPT to automate routine operations.

Model Problem 5. We want to find the external IPs that are probably the management and control consoles, so we will look for repeated connections throughout the day between the two IPs. Follow these steps:

- Filter connections to external IP addresses.
- Extract connection time excluding minutes.
- Group IP connections by source and destination.
- The grouping above counts the number of unique values at connection time, the sum of bits sent to the destination IP address, and counts the number of connections between those two IP addresses.

The result of the request is shown in Figure 7.

```
import pandas as pd
from datetime import datetime

# Load your dataset into a Pandas DataFrame
# Replace 'your_dataset.csv' with the actual path to your dataset
df = pd.read_csv('your_dataset.csv')

# Step 1: Filter connections to external IP addresses
external_connections = df[~df['source_ip'].str.startswith('192.168')]

# Step 2: Extract connection time excluding minutes
external_connections['timestamp'] = pd.to_datetime(external_connections['timestamp'])
external_connections['hourly_timestamp'] = external_connections['timestamp'].dt.floor('H')

# Step 3: Group IP connections by source and destination
grouped_connections = external_connections.groupby(['source_ip', 'destination_ip', 'hourly_timestamp'])

# Step 4: Count unique values at connection time, sum of bits sent, and count number of connections
result_df = grouped_connections.agg({
    'timestamp': 'nunique',
    'bits_sent': 'sum',
}).reset_index()

# Rename the columns for clarity
result_df.columns = ['source_ip', 'destination_ip', 'unique_connection_times', 'sum_bits_sent']

# Count the number of connections
result_df['num_connections'] = grouped_connections.size().reset_index(name='num_connections')['num_connections']

# Filter IPs with repeated connections throughout the day
probable_consoles = result_df[result_df['unique_connection_times'] > 1]

# Display the result
print(probable_consoles)
```

Figure 7: ChatGPT software code

	Source address	Destination address	Connection Time	Bytes Sent	Repeat Count
356	172.16.202.123	84		13 237042	3039
372	172.16.202.14	84		13 237198	3041
378	172.16.202.2	84		13 228696	2932
377	172.16.202.19	84		13 234312	3004
376	172.16.202.18	84		13 228852	2934
...
589	172.16.212.46	84		2 9380	134
587	172.16.212.45	84		2 15960	228
586	172.16.212.45	84		2 115398	26
576	172.16.212.31			2 10671	2
1	172.16.10.13	84.8		2 21549	21

Figure 8: The result of executing the program code

The result is a list of IP addresses that need to be analyzed, because the fact that there are thousands of connections between two IP addresses during 13 different periods of the day (according to the log) is suspicious activity, and can be classified as a potentially dangerous cyber incident.

We will search for long connections in the traffic. These types of connections typically need to be analyzed as more attackers use remote assistance tools like TeamViewer to avoid detection and maintain access to the organization's network. For this, we will use the following question:

Model Problem 6. We want to create a graph of long connections between an internal IP address and an external IP address by following the steps below.

- Filter messages to external IP addresses.
- Sum the connection time for each destination IP address and store it in a separate column called "sum_time".
- Add the number of connections made to each destination IP address and store it in a separate "sum_conn" column.
- Filter and store results with 'sum_conn' greater than 10.
- Keep only one line for each destination IP address.
- Divide the number of connections by the sum of connections and store it in a column called "avg_conn".
- Filter the ten results with the highest "avg_conn" value.
- Create a graph using the matplotlib library, where the "x" axis is the number of connections and the "y" axis is the total connection time.

The response of the artificial intelligence system to the request is shown in Fig. 9.

```
# Filter communications to external IP addresses
external_ips = data[data['Destination address'].str.startswith(('10.', '172.', '192.)) == False]

# Sum connection time to each destination IP and save it in a new column 'sum_time'
external_ips['sum_time'] = external_ips['Elapsed Time (sec)'].groupby(external_ips['Destination address']).transform('sum')

# Sum number of connections to each destination IP and save it in a new column 'sum_conn'
external_ips['sum_conn'] = external_ips['Destination address'].groupby(external_ips['Destination address']).transform('count')

# Filter duplicated values in 'Destination address'
external_ips = external_ips.drop_duplicates(subset='Destination address', keep="first")

# Filter results with 'sum_conn' greater than 10
external_ips = external_ips[external_ips['sum_conn'] > 10]

# Calculate average connections and save it in a new column 'avg_conn'
external_ips['avg_conn'] = external_ips['sum_time'] / external_ips['sum_conn']

# Filter the top 10 results with highest 'avg_conn'
top_results = external_ips.sort_values('avg_conn', ascending=False).head(10)

# Create a graph with matplotlib
import matplotlib.pyplot as plt

plt.scatter(top_results['sum_conn'], top_results['sum_time'])
plt.xlabel('Number of connections')
plt.ylabel('Total connection time')
plt.show()
```

Figure 9: ChatGPT software code.

When this program code is executed, we get the following result shown in Figure. 10.

Destination address	sum_time	sum_conn	avg_conn
293528	31	38663	46 840.500000
1364052	216.5	3504	13 269.538462
167133	3	9039	47 192.319149
1311770	142.250	1412	15 94.133333
2509298	31	1415	16 88.437500
2380559	142.250	1200	14 85.714286
2172240	3	1336	23 58.086957
112505	31	15286	304 50.282895
2166816	31	411	20 20.550000
115592	3	2026	111 18.252252

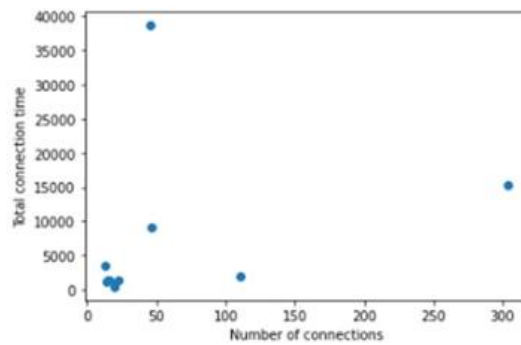


Figure 10: The result of executing the program code

3. Conclusions

Therefore, data analysis using AI is one of the key aspects of the future across virtually any field. However, in the case of cybersecurity, it becomes an essential skill for analysts. Knowledge of how to utilize tools such as LLM and artificial intelligence will impact the effectiveness of incident investigation and security monitoring. Cybersecurity is increasingly crucial in the modern world, and analysts must be prepared to employ contemporary methods and tools to safeguard data and networks.

4. References

- [1] Yanev Martin, «Building AI Applications with ChatGPT APIs», Published by Packt Publishing Ltd. (2023), 258 pp.
- [2] Caelen Olivier, Blete Marie-Alice «Developing Apps with GPT-4 and ChatGPT. Build Intelligent Chatbots, Content Generators, and More», Published by O'Reilly Media, Inc. (2023), 183 pp.
- [3] Chou, E. (2020) Mastering Python Networking. 3rd edn. Packt Publishing. Available at: <https://www.perlego.com/book/1365840/mastering-python-networking-your-onestop-solution-to-using-python-for-network-automation-programmability-and-devops-3rd-edition-pdf>
- [4] Mark Lutz, Learning Python, Fourth Edition. Published by O'Reilly Media, Inc., (2009), 1213 pp.
- [5] Kneusel Ronald T. «How AI Works: From Sorcery to Science» No Starch Press, (2023), 192 pp.
- [6] Jeremy Morgan, ChatGPT Vs Bard: Which is better for coding? URL: <https://www.pluralsight.com/blog/software-development/chatgpt-vs-bard-coding>
- [7] D. Khusainov, J. Diblik, A. Shatyko, J. Bastinec. Estimates of Solution Convergence Dynamical Processes in Neuronet with Time Delay // Conference Proceedings «IEEE ATIT 2019», p.411-414.
- [8] Andriy Shatyko, Denys Khusainov, Oleksii Bychkov, Josef Diblik and Jaromir Bastinec. Construction and Optimization of Stability Conditions of Learning Processes in Mathematical Models of Neurodynamics // CEUR Workshop Proceedings «IT&I-2022», 2022, Vol.3384, p. 42–51
- [9] Gartner Identifies the Top Cybersecurity Trends for 2023 Gartner Identifies the Top Cybersecurity Trends for 2023 URL: <https://www.gartner.com/en/newsroom/press-releases/04-12-2023-gartner-identifies-the-top-cybersecurity-trends-for-2023>
- [10] Dzhalladova, Irada & Ruzickova, Miroslava. (2020). Dynamical system with random structure and their applications». Cambridge Scientific Publishers, ISBN: 978-1-908106-66-7.
- [11] Kaminsky, O., Koval, V., Yereshko, J., Vdovenko, N., Bocharov, M., & Kazancoglu, Y. (2023). Evaluating the effectiveness of enterprises' digital transformation by fuzzy logic. *Advances in soft computing applications*, pp. 73–87.