ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

UDC 004.056.5:004.738 doi: 10.32620/reks.2024.3.09

Arkadii KRAVCHUK, Mykola ONAI

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

DEVELOPING INFORMATION TECHNOLOGY FOR EVALUATING AND ENHANCING APPLICATION-LAYER DDOS ATTACK DETECTION METHODS

*The **subject matter** of this article is the methods to detect distributed denial-of-service (DDoS) attacks at the Hypertext Transfer Protocol (HTTP) level with the purpose of justifying the requirements for creating software capable of identifying malicious web server clients. The **goal** of this article is to develop an information technology to evaluate the efficiency of DDoS attack detection methods, which will quantify their operating time, memory consumption, and approximate classification accuracy. In addition, this paper proposes hypotheses and a potential approach to improve existing application-layer DDoS attack detection methods with the intention of increasing their accuracy and identification speed. The **tasks** of this study are as follows: to analyse modern methods for detecting application-layer DDoS attacks; to investigate their features and shortcomings; to develop a software system to assess DDoS attack detection methods; to programmatically implement these methods and experimentally measure their performance indicators, specifically: classification accuracy, operating time, and memory usage; to compare the efficiency of the investigated methods; to formulate hypotheses and propose an approach to improve existing methods and/or develop new methods based on the results obtained. The **methods** employed are abstraction, analysis, systematic approach, and empirical research. In particular, the datasets generated by DDoS utilities were processed using the synthetic minority oversampling technique (SMOTE) to balance them. Furthermore, the studied DDoS attack detection methods were implemented, including fitting the required parameters and training artificial neural network models for evaluation. The following **results** were obtained. The average classification accuracy, operating time, and random-access memory (RAM) consumption during Internet traffic classification were determined for six DDoS attack detection methods under the same conditions. This study has demonstrated that the development of a novel method to detect DDoS attacks at the HTTP level with enhanced accuracy and classification speed is strongly required. The experimental results demonstrate that the time series-based method exhibited the shortest operating time (1.33 ms for 5000 vectors), whereas the deep neural network-based method exhibited the highest average classification accuracy (ranging from 99.07% to 99.97%) and the lowest memory consumption (39.09 KB for 5000 vectors). **Conclusions**. In this study, a software system was developed to assess the average accuracy of DDoS attack classification methods and measure the computational resources utilized. The scientific novelty of the obtained results lies in the formulation of two hypotheses and a potential approach to the creation of a novel method for detecting DDoS attacks at the HTTP level, which will have both high classification accuracy and a short operating time to surpass previously studied analogues in these respects. The first hypothesis is based on the additional usage of HTTP request attributes during Internet traffic classification. The second hypothesis is to analyse a graph of user transitions between website pages. The article also superficially describes a potential approach that involves the implementation of the described hypotheses as well as the proposed software architecture of an application-layer DDoS attack detection system for the Kubernetes platform and the Istio framework, which addresses the issue of collecting web request parameter values for websites that use the cryptographically secured HTTPS protocol.*

Keywords: *DDoS; DDoS attack detection; network traffic analysis; information security; AL-DDoS; HTTP; cryptography; software system; Kubernetes; Istio.*

Introduction

Motivation

The availability and reliability of Internet services are prerequisites for successful completion of most tasks in various areas of everyday life. For example, card payments in a shop require transaction processing on bank servers, while distance education is carried out using video conferencing systems and online learning platforms. Indeed, the modern development of information technology has made it possible to automate a multitude of routine tasks, rapidly process vast arrays of data, identify optimal solutions for programmed tasks, manage complex systems in an automated manner, and so forth. In many cases, the success of these scenarios

depends on the stable operation and availability of web servers that execute relevant software and serve user requests. Moreover, the stability of an Internet connection and data transfer speed play crucial roles in the field of information and communication technologies, as they facilitate remote provision of services to any location worldwide.

A disruption in the functioning of an online service, in addition to causing inconvenience to customers, may result in significant losses to the owners of the service, including financial and reputational losses, reduced productivity, and data privacy breaches. Therefore, ensuring the stability of servers providing online services is an important task in the information technology field, which involves protecting against various types of cyberattacks. It is also important to note that cyberattacks have become highly effective weapons of aggression in the context of contemporary hybrid wars [1]. Such attacks can disrupt the functioning of critical infrastructure and even worsen the moral and psychological state of the population, as is unfortunately true in Ukraine. In the context of a full-scale war, the stability of the computer systems used by civilians or defence forces is critical because cyber threats only increase in such circumstances.

There are numerous types of cyberattacks that aim to make Internet services unavailable to users, but researchers distinguish a separate type of cyberattack for this purpose – denial-of-service attacks, among which distributed denial-of-service (DDoS) attacks are the most well-known [2]. Furthermore, among all types of cyberattacks, DDoS attacks are among the most prevalent causes of server outages [3]. This is because such attacks do not incur high costs or special preparation, and the damage caused is quite significant. It should also be noted that in Ukraine during the third quarter of 2023, the proportion of DDoS attacks in all cyberattacks was approximately 90% [4]. Given the growing number of Internet of Things (IoT) devices that can be easily hacked by attackers to be used in distributed attacks, the issue of DDoS protection is becoming increasingly critical. In addition, as cloud computing continues to develop rapidly, an increasing number of companies will likely use it to provide services to customers via the Internet. The preceding argument leads us to conclude that the number of Internet services and cyberattacks will continue to increase. Thus, protecting computer resources from DDoS attacks is highly relevant.

DDoS attacks are designed to overwhelm the victim server with an excessive number of requests; thus, the main way to protect against such attacks is to filter incoming traffic from malicious users. It is evident that detecting malicious requests is significantly more complex than blocking network packets by IP address, so this article will focus on DDoS detection methods.

State of the art

There are numerous types of DDoS attacks, and for systematization, scientists have divided them into categories. One such categorization distinguishes between attacks in the network and application layers of the Open Systems Interconnection (OSI) model. Over the past few decades, scientists have conducted in-depth analyses of the characteristics of Internet traffic during network-layer DDoS attacks and have proposed effective methods to identify such traffic. For example, Ohsita et al. [5] described a method for detecting SYN flood attacks using statistical analysis. They justified that the rate of TCP SYN packets from benign users has a normal distribution and proved that high variance values for this rate can be a sign of a DDoS attack. Furthermore, Bogdanoski et al. [6] proposed an adaptive threshold algorithm for the detection of such attacks. This algorithm calculates the number of TCP packets with the SYN flag set and compares it with a threshold value that can vary depending on the total number of packets received over a given period. In addition, Boro et al. [7] considered a method to detect UDP flood attacks, which first counts the number of changes in the destination port and source IP address across packets originating from clients and then computes the Renyi entropy based on these obtained values. When an anomalous entropy value exceeds a predetermined threshold range, it is interpreted as an indicator of a potential attack. In other words, there are many methods for detecting network-layer DDoS attacks that are highly accurate and offer high classification speed. Moreover, the success of defence against network-layer attacks depends on the distribution of data centres involved and their bandwidth capacity to withstand heavy loads. In contrast, application layer attacks have a quite different situation.

Experts have highlighted that application-layer DDoS attacks are challenging to identify [8] because they behave similarly to user traffic; therefore, the speed of sending requests is no longer a reliable indicator for detecting this type of attack, in contrast to network-layer attacks. Although application-layer DDoS attacks have recently been actively studied by researchers, and appropriate methods for their identification have begun to emerge, there is still a gap in the understanding of both the characteristics and features of these attacks and effective methods for defending against them. Therefore, this article focuses on application layer attacks because existing methods either cannot detect them with high accuracy or take a long time to determine them. Regarding the gaps in this subject area, it is also worth noting that authors frequently fail to provide information regarding the operating time and memory consumption of developed solutions in their publications.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Thus, research into the detection of application-layer DDoS attacks is highly relevant. Analysis of modern methods for identifying these cyberattacks, in conjunction with experimental evaluation of the time and memory utilization of the corresponding software implementations during their operation, is essential to identify shortcomings and objectively compare the efficiency of existing solutions. This will facilitate the formulation of new approaches intended to enhance the detection process of application-layer DDoS attacks.

The paper structure

The remainder of this paper is divided into six sections and conclusions. The structural organization of the article is given below.

In **Section 1**, the main purpose of this study is outlined, and all objectives, which were fulfilled during this research, are enumerated in detail.

Section 2 presents a comprehensive literature review of modern methods for detecting application-layer DDoS attacks, providing an in-depth analysis of the current state-of-the-art techniques. In this section, six relevant methods are examined and their respective advantages and disadvantages are discussed, followed by a comparative analysis.

Section 3 describes the methodology employed during the experiment to assess the efficiency of the analysed DDoS attack detection methods. This section describes the approach used to generate datasets, the tools used to implement selected methods and develop the test software system, and the quantitative measures used to evaluate the accuracy, operating time, and memory usage of the considered methods.

Section 4 presents the case study of this paper by describing the experimental evaluation process of the analysed methods and the results of the conducted experiments. This section explains the dataset generation workflow and how the developed programs process data to prepare balanced datasets. In addition, it describes the software implementation of the system, which assesses the accuracy, speed, and memory consumption of the implemented DDoS attack detection methods during classification.

Section 5 discusses the obtained results. This section analyses the results of each method across all metrics and highlights their strengths and limitations. Furthermore, in this section, the methods are described in detail, including the selected optimal parameter values.

In **Section 6**, hypotheses and an approach to improve DDoS attack detection methods are proposed based on the analysis of the results of the considered methods. This section outlines the architecture of the software system for detecting application-layer DDoS attacks and describes the developed approach.

The paper ends with the **Conclusions**, which summarize the findings and highlight the scientific novelty and practical significance of this research. In addition, future research directions are outlined.

1. Purpose and objectives

The purpose of this study was to develop an information technology to evaluate the efficiency of DDoS attack detection methods. The technology will quantify the time and memory consumption of these methods and provide an approximate estimation of their average accuracy. Additionally, based on the analysis of the quantitative results obtained for existing methods, this study aims to propose an approach to improve application-layer DDoS attack detection methods with the objective of increasing their accuracy and/or detection speed.

To achieve this purpose, the following objectives must be performed: to analyse modern methods of detecting application-layer DDoS attacks and investigate their features and shortcomings; to develop a software system to assess DDoS attack detection methods; to programmatically implement the considered methods and experimentally measure their performance indicators (classification accuracy, operating time, memory usage); to compare the efficiency of the investigated methods; and to formulate hypotheses and propose an approach for improving existing methods and/or developing new methods based on the results obtained.

2. Application-layer DDoS attack detection methods

It is necessary to analyses the specifics of this subject area and its terminology. As previously stated, DDoS attacks are categorized into two distinct classes according to the OSI model employed: network-layer (NL-DDoS) and application-layer (AL-DDoS) attacks. In fact, DDoS attacks utilize a specific communication protocol, such as ICMP, TCP, UDP, DNS, HTTP, and others, according to which the affiliation to the corresponding class is determined. However, regardless of the protocol used, each class has its own specific characteristics: NL-DDoS attacks aim at overloading the bandwidth of communication channels, switches, and other network devices by sending an excessive number of packets per unit of time, whereas AL-DDoS attacks aim at exhausting the computing resources of the server that processes the requests. In addition, the following types of DDoS attacks are distinguished by the speed of malicious traffic generation: high-rate and low-rate. The majority of NL-DDoS attacks are high-rate (HR-DDoS) attacks because modern data require an extremely large amount of data to overload them. This is achieved by

continuously sending network packets from numerous devices simultaneously with minimal delays, i.e. the transmission speed under such conditions is very high. However, AL-DDoS attacks do not require sending a substantial amount of data at high speed because the target server can be overwhelmed by a relatively small number of special requests, which nevertheless require a considerable amount of computational resources (e.g., CPU time or RAM) for processing. This is the reason why application-layer attacks are commonly low-rate (LR-DDoS), which significantly complicates the detection process [9]. For example, uploading large files to a server involves a resource-intensive write operation on disk, yet this activity often appears to be a typical client behaviour. Thus, the peculiarity of detecting application-layer DDoS attacks lies in focusing more on the content of requests and their attributes when analysing Internet traffic data rather than on the speed at which requests are received. To gain a more comprehensive understanding of the classification of DDoS attack types, Fig. 1 shows a diagram that organizes the types of attacks into several categories.

The identification of DDoS attacks is a binary classification task, whereby each sender of web requests should be assigned to one of two classes: benign or malicious user. To determine the appropriate class, network traffic data are collected, processed, and analysed. This information is then used to identify the sources of DDoS attacks using a specific classification method. Cybersecurity experts have identified two distinct categories of DDoS detection methods: those based on signature detection and those based on anomaly detection [10]. Furthermore, the latter category encompasses a number of additional approaches that employ statistical analysis, information theory, artificial neural networks, and other techniques. It is evident that algorithms that search for matches between traffic attributes and attack signatures in a database can identify attacks almost instantly. However, signature-based methods are not suitable for detecting AL-DDoS attacks because such attacks do not have stable attribute values that can be used to identify them. In contrast, anomaly-based methods can detect even new types of attacks, but require more time for classification. Consequently, this paper considers anomaly-based methods, which perform more in-depth data analysis based on certain mathematical techniques and are able to comprehensively classify application-layer DDoS attacks. A clear representation of the classification of DDoS attack detection methods is provided in Fig. 2, which shows the hierarchical structure of these methods, dividing them into categories and types.

No et al. [11] proposed a method for detecting DDoS attacks using information theory to reduce the computational complexity of calculating the information entropy of the rate of incoming network packets from users. The primary advantage of the Fast Entropy method is its ability to accelerate the detection of DDoS attacks, thereby enabling their rapid blocking and mitigation. In addition, No et al. [11] provided experimental evidence that their Fast Entropy method has the shortest identification time and the highest classification accuracy among existing methods based on information entropy. However, the main limitation of the proposed algorithm is that it was evaluated only in the context of HR-DDoS attacks at the network layer. Therefore, the proposed method cannot detect application layer attacks due to its reliance on a single network layer feature.

In contrast to No et al. [11], Zhao et al. [12] selected other attributes to analyse user behavior on websites and used such attributes to detect AL-DDoS attacks. Similarly to the previously mentioned method, this method is based on information entropy but employs other formulas for new attributes. Specifically, it utilizes the Uniform Resource Locator (URL) of the requested resource and the client's IP address as attributes to calculate two entropy values for each unique client and web server resource. The entropy of URL resources requested by a given client's IP address is as follows:

$$\mathrm{EUPI}(\mathrm{i}) = \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{P}_{\mathrm{k}}^{\mathrm{URL}}(\mathrm{x}_{\mathrm{i}}) \log_2\left(\mathrm{P}_{\mathrm{k}}^{\mathrm{URL}}(\mathrm{x}_{\mathrm{i}})\right), \qquad (1)$$

where $\mathrm{EUPI}(\mathrm{i})$ is the information entropy of URLs from the i^{th} IP address x_{i};

$\mathrm{P}_{\mathrm{k}}^{\mathrm{URL}}(\mathrm{x}_{\mathrm{i}})$ is the probability of occurrence of the k^{th} URL in requests from the i^{th} IP address x_{i};

K is the number of unique URLs in requests from the i^{th} IP address x_{i}.

Formula (1) can be used to identify a DDoS attack on a specific web server resource. However, during periods of high user activity, such as a flash crowd event where a large number of users visit, for example, a promotional product page in an online store, this indicator may falsely signal a DDoS attack. Therefore, the authors proposed another formula for calculating the entropy of IP addresses that have accessed a particular URL resource on the server:

$$\mathrm{EIPU}(\mathrm{k}) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{P}_{\mathrm{i}}^{\mathrm{IP}}(\mathrm{u}_{\mathrm{k}}) \log_2\left(\mathrm{P}_{\mathrm{i}}^{\mathrm{IP}}(\mathrm{u}_{\mathrm{k}})\right), \qquad (2)$$

where $\mathrm{EIPU}(\mathrm{k})$ is the information entropy of IP addresses for the k^{th} URL u_{k};

$\mathrm{P}_{\mathrm{i}}^{\mathrm{IP}}(\mathrm{u}_{\mathrm{k}})$ is the probability of occurrence of the i^{th} IP address in requests to the k^{th} URL u_{k};

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

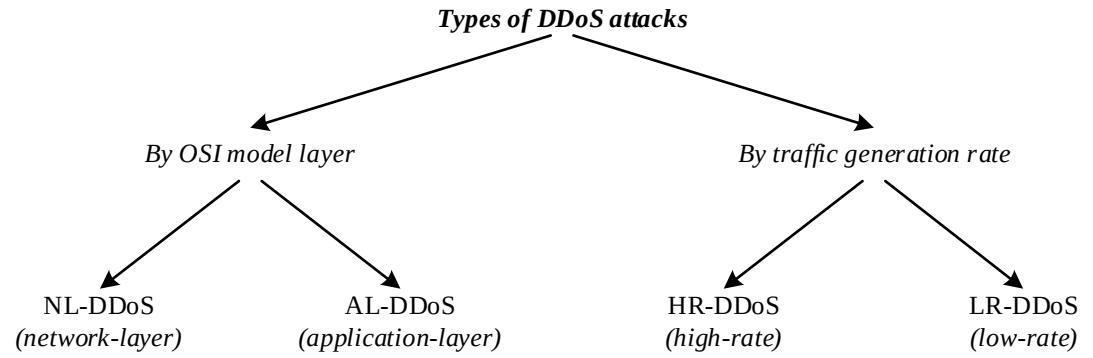

Fig. 1. Classification scheme for DDoS attack types

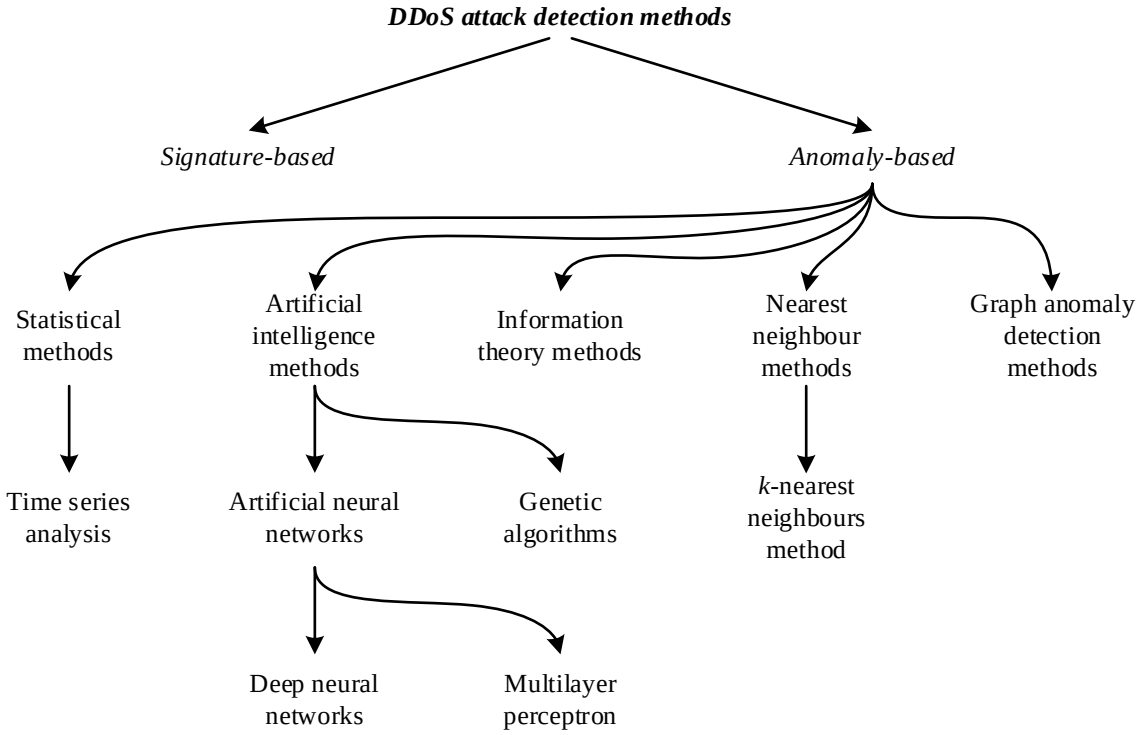

Fig. 2. Classification scheme of DDoS attack detection methods

N is the number of unique IP addresses in requests to the k^{th} URL u_k.

It is argued that the proposed method can accurately detect HTTP DDoS attacks on web servers and distinguish them from flash crowd events by checking the entropy of IP addresses that accessed a URL resource with suspicious activity using the formula (2). However, this approach does not consider other attributes of web requests that could cover a wider range of attacks against web server resources. Furthermore, it is unclear how DDoS attack detection software can obtain specified attributes from user requests using an HTTPS protocol, which encrypts all data between the server and client.

Laptyev et al. [13] proposed a method to detect LR-DDoS attacks executed via slow HTTP requests. The proposed method leverages time series analysis to predict future user behavior, thereby allowing for the pre-emptive classification of all site visitors and blocking suspicious ones. A key attribute is the delay time between packets received from the client, which is quantified using the following formula:

$$T = \frac{\sum_{i=1}^{k}(t_{i+1} - t_i)}{k-1}, \quad (3)$$

where T is the average delay between received packets;

k is the number of packets received for analysis;

t_{i+1} is the time of receipt of the $i+1^{st}$ packet;

t_i is the time of receipt of the i^{th} packet.

The construction of this time series involves analysing the time intervals between successive user packets using the formula (3), and the focus of the method is to identify abnormally large delays or a trend toward increasing delays, which is indicative of an LR-DDoS attack. One disadvantage of this method is that analysing the time series data for each website visitor is time-consuming, potentially requiring a separate high-performance computer. A further limitation of this study was that it only analysed the time delay between packets. In addition, the authors of this method did not provide any conclusions regarding the optimal threshold value for packet delay that should signal a DDoS attack.

Dong et al. [14] proposed a method to detect DDoS attacks based on the k-nearest neighbours (KNN) algorithm and proposed an improvement to this method. The KNN method classifies network packets using their attributes, represented as a vector $(f_1(x_1), f_2(x_2), ..., f_n(x_n))$. This vector contains information about the number of packets, duration of receiving all packets, total number of received bytes, and average number of received bytes per second. This representation creates a Euclidean space, allowing the calculation of distances between packet vectors. As a result, by analysing the nearest neighbors of a packet being classified, we can determine whether it is benign or malicious. To enhance the classification precision, the authors of this method improved the KNN model by introducing an additional weighting factor into the formula to determine the class of an object as follows:

$$w = \frac{1}{e^{d(x_p, x_i)}}, \quad (4)$$

where w is the weight factor in the formula used to determine the class of the object;

$d(x_p, x_i)$ is the distance between the object under classification x_p and the current object x_i, which is selected for comparison with the given data set.

The application of formula (4) yielded a commendably high classification accuracy of 91%. Another noteworthy aspect of this method is its ability to gather attributes in software-defined networking (SDN) through the API of network controllers. However, a significant limitation of this method is its requirement for preliminary training using a dataset that provides comprehensive information about common types of application-layer DDoS attacks. In addition, this study does not specify how to determine the optimal value of parameter k, which determines the number of nearest objects considered when classifying a packet. In addition, the proposed solution has yet to be evaluated for its efficacy in detecting low-rate attacks.

Johnson Singh et al. [15] presented a method to classify HTTP DDoS attacks that employs artificial neural networks (ANN), specifically, the multilayer perceptron (MLP) and genetic algorithm (GA). The structure of the proposed neural network comprises three layers: an input layer, a hidden layer, and an output layer. Each layer contains three neurons, with the exception of the output layer, which contains only one node. This method distinguishes itself from other ANN-based approaches by using a genetic algorithm to initialize weights during model training, as opposed to the conventional use of gradient descent during the backward error propagation phase. Johnson Singh et al. [15] proposed using three attributes as inputs to the neural network: the number of HTTP requests, entropy, and variance of requests within each user session. Data collection was conducted over a recurring 20 s interval, which, according to the authors, was sufficient for a DDoS attack to make a website inaccessible. It is also important to note that in this method, only HTTP GET requests are considered for analysis. The entropy of HTTP GET requests during the session of the i^{th} user within the w_t timeframe is:

$$E_{(i,w_t)} = -\log \frac{C_{(i,w_t)}}{\sum_{j=1}^{n} C_{(j,w_t)}} + \left| \log \left(\frac{C_{(i,w_{t+1})}}{C_{(i,w_t)}} \right)^{(-1)^{\lambda}} \right|, \quad (5)$$

where $E_{(i,w_t)}$ is the information entropy of the web requests of the i^{th} user within the time interval w_t;

$C_{(i,w_t)}$ is the number of requests received from the i^{th} user within the time interval w_t;

n is the number of unique users captured within the time interval w_t;

λ is the parameter whose value is 0 when $C_{(i,w_t)} \geq C_{(i,w_{t+1})}$, or 1 when $C_{(i,w_t)} < C_{(i,w_{t+1})}$.

Upon calculating the entropy indicators of user web requests using formula (5) over several time intervals, the variance of these entropy values is obtained as follows:

$$V_i = \frac{\sum_{j}^{N} \left(E_{(i,w_j)} - M_i \right)^2}{N}, \quad (6)$$

where V_i is the variance of the entropy values of the web requests of the i^{th} user;

M_i is the mean value of the entropy values of the web requests of the i^{th} user;

N is the number of time intervals within which all entropy values for the i^{th} user are obtained.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Formulas (5) and (6) are used to calculate the values of the key attributes which characterize the client's behaviour during communication with the HTTP server. The proposed method can detect application-layer DDoS attacks on web servers with a high degree of accuracy, and the detection rate was approximately 98% based on the experimental results. Furthermore, the proposed method can distinguish between DDoS attacks and flash crowd events by comparing the variance in the entropy values of web requests and the total number of requests for each suspicious user. However, the proposed method has certain limitations. It is only capable of analysing GET requests, which represents a significant restriction because numerous HTTP DDoS attacks employ other web request methods, such as POST, PUT, and PATCH. Another drawback of this method is that the proposed attributes are based solely on the number of user requests. Consequently, although the proposed method can identify LR-DDoS attacks generated by Slowloris and SlowHTTPTest, it is unlikely to be able to detect other types of low-rate DDoS attacks. In addition, classification accuracy depends on the training dataset that contains sample requests from various types of HTTP DDoS attacks.

Muraleedharan et al. [16] proposed a deep neural network (DNN) model to identify slow DDoS attacks at the HTTP protocol level. The choice of DNN in the given study is justified by the fact that, among all artificial neural networks, this type has one of the highest data classification accuracy rates. Furthermore, deep neural networks have gained significant popularity, as evidenced by the rapid development of large language models (LLMs), which are constructed using DNN architectures and have demonstrated remarkable capabilities in natural language processing tasks. The difference between a deep neural network and a conventional ANN (e.g., MLP) is that a DNN has multiple hidden layers with neurons rather than a single layer. In addition, a DNN is a fully connected feed-forward neural network; thus, each neuron is connected to all neurons in the subsequent layer. This allows us to recognize nonlinear relationships between data. The DNN model proposed in this study incorporates four hidden layers, with each layer containing the same number of neurons as the input layer. The input data for this model comprise a set of 80 attributes from the flow of network packets, including protocol type, number of received bytes, packet count, and transmission speed. Thus, the input and hidden layers contain 80 neurons. In order to train the model, the open dataset CICIDS2017, which contains all of the aforementioned attributes, was employed, as well as a class label for each packet. This label indicates benign traffic or one of the four DDoS tools: Slowloris, SlowHTTPTest, Hulk, and GoldenEye. The main feature of the proposed method is that it performs multi-class classification, which enables determination of the name of the program that generated the detected DDoS attack. Thus, the output layer of the DNN comprises five neurons, each responsible for a distinct class: benign traffic and four DDoS utilities. Among the aspects of the implementation of this neural network, the following should be noted: ReLU (rectified linear unit) and Softmax activation functions were used for the hidden layers and the output layer, respectively; the Adam (adaptive moment estimation) optimization algorithm was used to train the model; and the categorical cross-entropy loss function was chosen for classification. The proposed method exhibits a remarkably high classification accuracy rate of nearly 99%, which significantly outperforms the existing methods. However, the accuracy of the proposed method was derived from a relatively small dataset comprising only 6000 attribute vectors, with only 2000 representing benign traffic. The limited sample size raises concern about the practical applicability of the findings. To obtain more objective results, it is advisable to employ a balanced dataset and a more substantial and diverse data collection to thoroughly evaluate the proposed method. Although this study did not measure the classifier's operating time, given the complexity of DNN and the high dimensionality of the data, it is fair to say that this method has a relatively low attack detection rate compared to the other analysed methods. In the DDoS attack detection context, any delay is a significant disadvantage because it is important to identify the attack before it makes the website unavailable to users. Another challenge in implementing this method is the considerable time required to train the model and the necessity to find a balanced dataset with different types of HTTP DDoS attacks.

This section presents a comprehensive analysis of six DDoS attack detection methods, evaluates their advantages and weaknesses, pinpoints their limitations, and explores their potential use in identifying DDoS attacks at the application layer. In addition, a set of key criteria was established to compare the abovementioned methods (Table 1).

According to Table 1, the EUPI-EIPU [12], Time series [13], MLP-GA [15], and DNN [16] methods can be employed to detect application-layer DDoS attacks. In addition, the KNN [14] method can be used if new attributes that characterize AL-DDoS attacks are selected, and in such cases, the model is trained on a dataset containing the corresponding features. In addition, almost each of the above methods is capable of identifying LR-DDoS attacks; however, the authors have not conducted exhaustive experimental trials to test their proposed solutions against different types of attacks. For example, the authors of the MLP-GA [15] method did not assess the classification accuracy of low-rate DDoS

Table 1

Comparative characteristics of DDoS attack detection methods

Criteria / Method	OSI model layers; protocols	Attributes employed	Requirements for usage		Type of DDoS attacks			
			supervised learning	parameters tuning	NL	AL	HR	LR
Fast entropy [11]	3, 4; ICMP, TCP, UDP	speed of receiving packets	–	+	+	–	+	–
EUPI-EIPU [12]	7; HTTP	URL of the request, IP address of the client	–	+	–	+	–	+
Time series [13]	4, 7; TCP, HTTP	delay time between packets	–	+	–	+	–	+
KNN [14]	3, 4; ICMP, TCP, UDP	number of packets and bytes, speed and duration of receiving bytes	+	+	+	–	+	–
MLP-GA [15]	7; HTTP	number of requests, their entropy, variance	+	–	–	+	+	–
DNN [16]	4, 7; TCP, HTTP	properties of network packet flow	+	–	–	+	–	+

attacks. In addition, the number of requests does not adequately characterize LR-DDoS attacks, it is advisable to use other attributes in entropy calculations when using this method. Therefore, selecting the optimal attributes to detect low-rate AL-DDoS attacks is an important task. Upon examination of the data presented in Table 1, it is evident that the HTTP is the most prevalent application-layer attack protocol. It is also important to note that all methods require preliminary preparation prior to use, including finding thresholds for correctly signalling an attack, selecting parameters (e.g., k for KNN), and training a model for machine learning methods.

In order to draw final conclusions about the analysis of modern methods for detecting application-layer DDoS attacks, it is necessary to compare the classification accuracy, operating time, and memory consumption of these solutions under the same experimental conditions, specifically, with the same set of test data, the same amount of free computing resources of the same computer, and the same programming language for simple meeting software components and their dependencies.

3. Methodology

The methodology of the experiment to determine the efficiency of the analysed AL-DDoS attack detection methods requires discussion. This step is necessary to objectively evaluate the considered solutions in a common environment using the same test data. The experimental study consists of three parts: preparation of the dataset, implementation of the methods and software system for testing, and quantitative evaluation of the method efficiency.

The purpose of the dataset preparation phase is to generate a dataset comprising network packets typical of both benign and malicious traffic, which are subsequently employed to train models and verify the accuracy of the classification methods. Because the available public datasets contain a limited number of examples of HTTP LR-DDoS attacks [17], we decided to generate a new dataset. This involves preparing a web server, finding utilities to simulate relevant DDoS attacks, capturing the generated network traffic, processing the captured data according to the requirements of the classifiers, and saving the resulting data in a structured format. To generate low-rate DDoS attacks at the HTTP protocol level, several utilities have been selected, each differing in their method of generating malicious traffic [18]. Specifically, the following programs are involved: Rudy [19], Slowloris [20], SlowHTTPTest [21], PyDDoZ [22], Hulk [23], and GoldenEye [24], most of which generate LR-DDoS attacks. The victim is a web server serving a

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

website with multiple pages (URLs) using the HTTP protocol, which provides web request attributes in an unencrypted format, allowing them to be obtained directly from the network packet. This server is launched from the source code [25] on a local computer, where DDoS attacks are generated.

To obtain user traffic, it is proposed to manually visit website pages. However, the number of such visits will be too small; thus, a special algorithm should be employed to increase the number of minority class examples to correct the dataset imbalance. It is also crucial that such algorithm should be responsive to non-stationary processes of traffic changes, thereby ensuring that synthetic data generated accurately reflect the dynamic nature of network traffic. However, given that each dataset in this study contains only a single manually generated instance of a DDoS attack without any additional concurrent events, the requirement for this algorithm to adapt to fluctuating traffic patterns is not applicable in this case.

Additionally, it is necessary to develop a program that analyses and processes the captured network packets to select and store the necessary attributes in a structured format. The Python programming language should be used to create such a program because it offers many libraries for processing network data (e.g., CICFlowMeter [26] and pyshark [27]), which significantly simplifies the process of searching and selecting network packet attribute values. Moreover, software development using Python is fast because of its concise syntax. Despite the low execution speed of Python programs, primarily due to the use of an interpreter, this is not a significant disadvantage in this case because the primary goal in this case is to obtain processed data.

The analysed methods for detecting DDoS attacks, as well as the software system for testing, were implemented in the Go programming language since it is important to obtain values for the operating time of the corresponding solutions under the same conditions. Go is a compiled programming language, so the execution speed of developed programs is high and comparable to other compiled languages. In contrast, Python is relatively slow. Although Python offers numerous packages for data processing and machine learning, these often execute calls to compiled libraries written in C, which compromises the objectivity of comparing different methods. To implement the software system, a strategy design pattern was selected to facilitate the convenient interchange of different DDoS attack detection methods during testing. Furthermore, methods requiring supervised learning are trained on a training dataset, which is a subset of a previously prepared dataset. In addition, the training dataset is used to select optimal parameter values as required.

A quantitative evaluation of the methods was conducted by measuring the following indicators: average classification accuracy, operating time, and RAM usage using the developed software. The efficiency of the methods was evaluated using a test dataset that was extracted from the main dataset generated during the first phase. In particular, the accuracy of DDoS attack detection methods is calculated as binary classification accuracy, which represents the proportion of correctly classified vectors to the total number of classified vectors. This approach only provides a general estimation of accuracy, which means that the resulting value is somewhat approximate and may not fully reflect the true accuracy of the method. This limitation arises because DDoS attacks are typically non-stationary processes, and the characteristics of traffic may vary over time. In dynamic network environments, short-lived DDoS attacks may go undetected, resulting in a small number of false negatives that do not significantly impact the overall accuracy score. Therefore, the accuracy score may be misleadingly high. However, in this study, each DDoS dataset was designed to contain only a manually generated DDoS attack, resulting in a more uniform traffic pattern. In addition, benign traffic was assumed to be evenly distributed throughout the attack duration, which simplified the evaluation process. Although this approach to calculate accuracy may not be suitable for more complex, real-world scenarios with multiple attacks and highly dynamic traffic, it is a reasonable approximation in the current controlled setting. Given these constraints, the goal of this study is not to determine the exact accuracy of DDoS attack detection methods; rather, we evaluate their performance under controlled conditions. Accordingly, binary classification accuracy was employed, and it was calculated using the following formula:

$$AC = \frac{TP + TN}{TP + FP + TN + FN}, \quad (7)$$

where AC is the average classification accuracy;

TP is the number of correctly classified attribute vectors of DDoS traffic;

TN is the number of correctly classified attribute vectors of benign traffic;

FP is the number of vectors incorrectly classified as DDoS traffic attribute vectors;

FN is the number of vectors incorrectly classified as benign traffic attribute vectors.

For all the analysed methods, it is necessary to calculate the average classification accuracy for each type of HTTP DDoS attack generated by the previously mentioned utilities using formula (7). Future research may investigate more comprehensive metrics and adaptive approaches for evaluating classification

accuracy, which would be better suited to varying traffic conditions and multiple attack scenarios. In addition, the operating time and RAM consumption of the developed software solutions should be measured several times using the built-in profiling tools in Go. The analysis should be based on the average values of these metrics to eliminate the impact of random factors.

It is important to note that the Fast entropy [11] method has not been experimentally evaluated, i.e., its accuracy, operating time, and memory consumption have not been measured. This is due to the fact that this method was designed for the specific purpose of detecting only network-layer DDoS attacks and relies on an attribute that characterizes only high-rate attacks.

4. Experimental evaluation of methods

During the preparation of the test dataset, several Bash scripts and Python programs were developed and employed to perform the necessary tasks. These tasks included deploying a web server, collecting Internet traffic parameters, generating DDoS attacks, processing network packets, and extracting the necessary packet and web request attributes for the analysed methods. To capture the traffic, the tcpdump utility was used to listen to all network interfaces and to save any network packets sent to or from the running web server. In other words, all incoming and outgoing traffic from the test website was monitored, and filtering was based on the port number of the sender or receiver matching the port number on which the website was running. Packets were captured during the execution of each DDoS attack generation tool for 5 min, except for Hulk [23] and GoldenEye [24], for which the capture duration was 5 s because these tools generate high-rate attacks. The traffic capture results were recorded in PCAP files.

The simulation of user traffic was conducted by manually opening website pages from three different browsers simultaneously and performing typical user actions such as registering on the site, authorization, creating and editing elements and uploading photos. To generate DDoS attacks, relevant utilities are launched from the corresponding source code [19 – 24]. Each utility was executed sequentially via scripts while the network traffic was captured, resulting in 6 PCAP files for the respective DDoS utilities and one additional PCAP file describing the user traffic. The settings used for the DDoS utilities are listed in Table 2. In addition, it should be noted that whenever possible, DDoS tools are configured to generate random URLs, headers, and request bodies.

Two distinct approaches were employed to process the obtained PCAP files with the objective of extracting the necessary attributes of Internet traffic from both the network and application layers. This was necessary because the Time series [13], KNN [14], and DNN [16] methods rely on the attributes of network packet flow, whereas the other analysed methods rely on information from HTTP requests. To calculate packet flow characteristics from the PCAP files, the CICFlowMeter [26] program was employed, which was used to create a dataset for training the DNN [16] model. The CICFlowMeter [26] program provides a total of 80 attributes about packet flow, including all necessary network layer attributes for the Time series [13] and KNN [14] methods. Finally, the calculated attributes are exported to CSV files. A Python program was developed and used to obtain information about HTTP requests. This program analyses the contents of packets in the PCAP file using the pyshark [27] library and extracts the parameters of each HTTP request from the TCP protocol data into a separate array even if the request is not fully completed. Subsequently, all captured web request attributes are written to a CSV file, including the IP address, port number, date and time the request was received, request method, URL, and the User-Agent of the request. Therefore, for seven different classes of captured data, the two programs produced 14 CSV files. The number of packets and requests from each Internet traffic source obtained as a result of the above data processing steps is shown in Table 3.

Since almost all of the DDoS detection methods analysed in this study perform only binary classification, and their identification accuracy can vary significantly depending on the type of attack, it was decided to create

Table 2

Parameters of utilities for generating DDoS attacks

Parameter / DDoS utility	Request methods	Request body size, KB	Delay between requests, s	Number of threads	Number of client sockets	Operating time, s
Rudy [19]	POST	5120	3	100	100	300
Slowloris [20]	GET	–	15	1	150	300
SlowHTTPTest [21]	GET	–	10	1	150	300
PyDDoZ [22]	GET and POST	–	0.4	128	128	300
Hulk [23]	GET	–	–	1022	1022	5
GoldenEye [24]	GET and POST	–	–	5	150	5

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Table 3

The amount of information after processing the generated traffic from various sources

Source or type of traffic	Number of obtained attribute vectors	
	at the network layer	at the application layer
Benign	67	492
Rudy [19]	346	100
Slowloris [20]	600	600
SlowHTTPTest [21]	430	210
PyDDoZ [22]	739	1484
Hulk [23]	2887	3008
GoldenEye [24]	2473	2471

datasets in which user traffic data are combined with data from each DDoS attack source for the network and application layers separately. This approach allows for a more accurate determination of the efficiency of these methods by classifying specific DDoS utilities, thereby identifying potential gaps in their implementation.

As shown in Table 3, combining benign traffic with any other source of DDoS attacks into a single dataset creates an unbalanced dataset, which typically results in inefficient training of machine learning models and, consequently, degrades classification accuracy. Therefore, another Python program was developed that combines information from two CSV files, one representing benign traffic, into one dataset and balances it by increasing the number of minority class values using the SMOTE [28] algorithm. However, it is important to acknowledge the limitations of using the SMOTE algorithm when balancing datasets containing network traffic data because it does not account for the non-stationary behaviour of DDoS attacks. Despite this limitation, the relatively stable network environment captured in the datasets, which were generated under controlled conditions in this study, allows for the utilization of more straightforward balancing approaches. Because benign traffic remains consistent throughout a single instance of a DDoS attack, the SMOTE algorithm remains a feasible choice for this study. However, in future research involving more dynamic network environments, it will be essential to develop a more advanced algorithm that can adapt to non-stationary traffic changes to ensure the reliability of the measured classification accuracy values of detection methods. Additionally, a dataset containing network traffic attributes from all sources of DDoS attacks [19 – 24] was generated and balanced for the DNN [16] method to evaluate the accuracy of the multi-class classification. As a result, 13 balanced datasets were created. The number of classes in each of the prepared datasets is given in Table 4.

For experimental evaluation of application-layer DDoS attack detection methods, a software system was developed in Golang, implementing the analysed methods [12 – 16]. The proposed system performs the following tasks: reading data from the generated datasets, training models or tuning parameter values for the corresponding methods, measuring their classification accuracy, operating time and memory usage, and exporting the obtained metric values to CSV files. Although Golang lacks the concepts of classes and inheritance, it supports interfaces, structures, and methods, thus enabling interface implementation features. In addition, Golang facilitates code reuse and modularity via the composition of structures.

The developed software system is the core component of the information technology designed to evaluate the efficiency of DDoS attack detection methods, and its primary workflow is illustrated in Fig. 3.

Table 4

Number of instances of each class in generated datasets

Dataset name	Class name in the dataset	Number of Internet traffic attribute vectors			
		at the network layer		at the application layer	
		by class	in total	by class	in total
1	2	3	4	5	6
Benign and Rudy [19]	Benign	346	692	492	984
	Rudy [19]	346		492	
Benign and Slowloris [20]	Benign	600	1200	600	1200
	Slowloris [20]	600		600	
Benign and SlowHTTPTest [21]	Benign	430	860	492	984
	SlowHTTPTest [21]	430		492	

Continuation of Table 4

1	2	3	4	5	6
Benign and PyDDoZ [22]	Benign	739	1748	1484	2968
	PyDDoZ [22]	739		1484	
Benign and Hulk [23]	Benign	2887	5774	3008	6016
	Hulk [23]	2887		3008	
Benign and GoldenEye [24]	Benign	2473	4946	2471	4942
	GoldenEye [24]	2473		2471	
Benign, Rudy [19], Slowloris [20], SlowHTTPTest [21], PyDDoZ [22], Hulk [23] and GoldenEye [24]	Benign	2887	20209	–	
	Rudy [19]	2887			
	Slowloris [20]	2887			
	SlowHTTPTest [21]	2887			
	PyDDoZ [22]	2887			
	Hulk [23]	2887			
	GoldenEye [24]	2887			

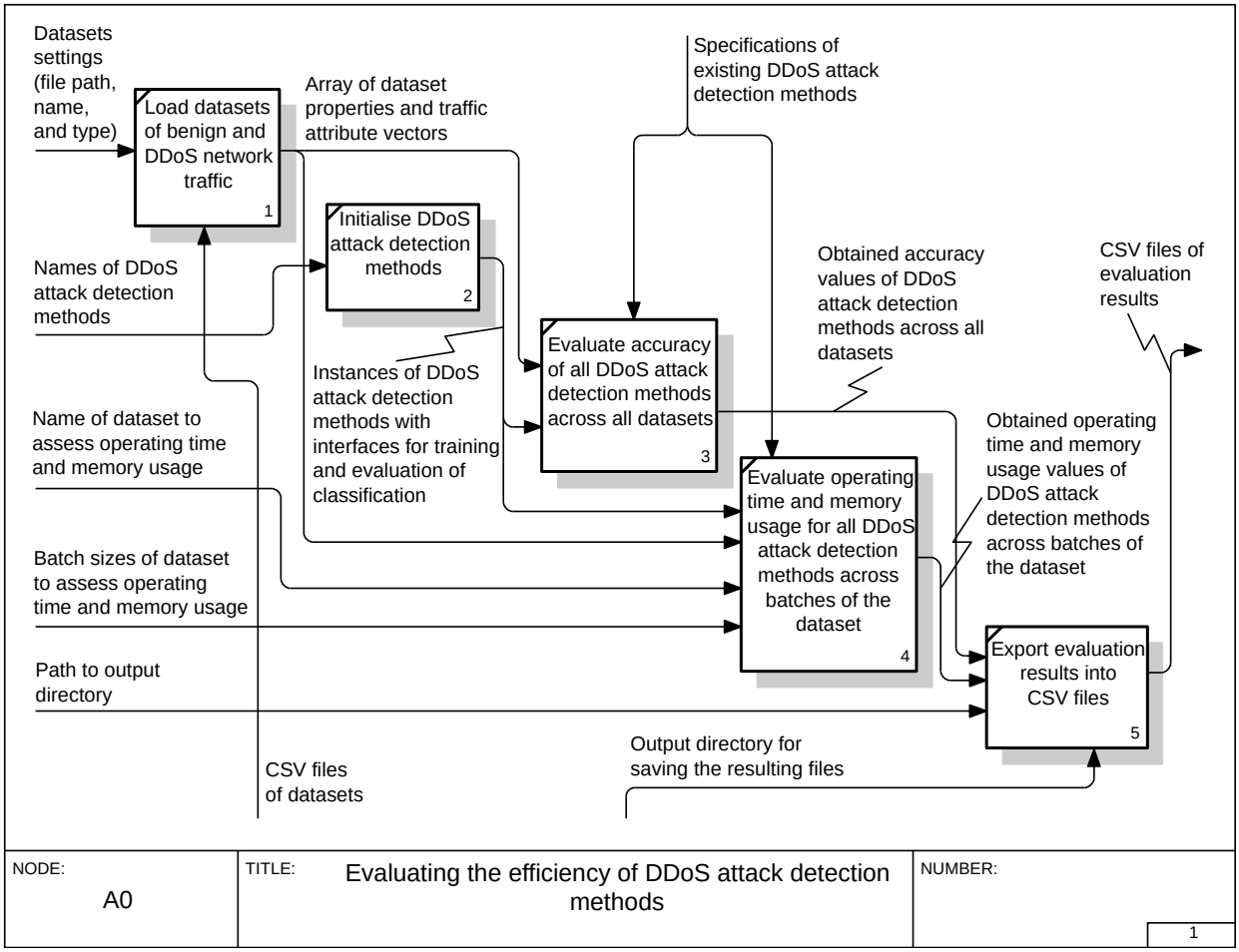

Fig. 3. IDEF0 diagram of information technology for evaluating the efficiency of DDoS attack detection methods

Fig. 3 shows an IDEF0 diagram that illustrates the processes involved in assessing DDoS attack detection methods, as well as their inputs, outputs, controls, resources, and interconnections. The main process (A0) is divided into five constituent steps: loading datasets, initializing detection methods, evaluating the accuracy of each method across all datasets, assessing their operating time and memory usage, and exporting the results to CSV files. This diagram provides a high-level overview of how the developed software system orchestrates these tasks in the overall information technology framework. It is important to note that the evaluation processes in

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

activities A3 (accuracy evaluation) and A4 (operating time and memory usage evaluation) are crucial and closely interrelated and are managed by the same module in the software system. Because of the similarities between these two processes, only the first is explored in depth. For a more detailed breakdown, the decomposition of process A3 is illustrated in Fig. 4.

Fig. 4 illustrates the IDEF0 decomposition of activity A3 from the top-level diagram in Fig. 3. This decomposition highlights that the process of evaluating the accuracy of DDoS attack detection methods is divided into five principal stages: selecting appropriate datasets for each method, splitting datasets into training, validation, and test subsets, training models of each method on all datasets, predicting traffic types for test subsets, and calculating classification accuracy using the predicted and expected values. The selection of an appropriate dataset is essential because this study utilizes two distinct types of datasets: one representing traffic data at the network layer and application layer. Each considered DDoS attack detection method is designed to operate with only one of these data types; thus, it is necessary to filter and match the supported datasets for each method during the initial step (A31). The model training process (A33) and traffic type prediction (A34) processes are implemented in accordance with the specifications of the respective DDoS attack detection methods, as detailed in Section 2 of this paper. The classification accuracy at the final step (A35) was calculated using the methodology outlined earlier in this article.

IDEF0 diagrams do not fully illustrate how the corresponding stages handle arrays of input data and other detailed operations, which are crucial for understanding the process. Therefore, Fig. 5 presents a flowchart of the algorithm to evaluate the accuracy, operating time, and memory usage of DDoS attack detection methods. The algorithm is implemented as a universal software module that is applicable to processes A3 and A4, as shown in Fig. 3.

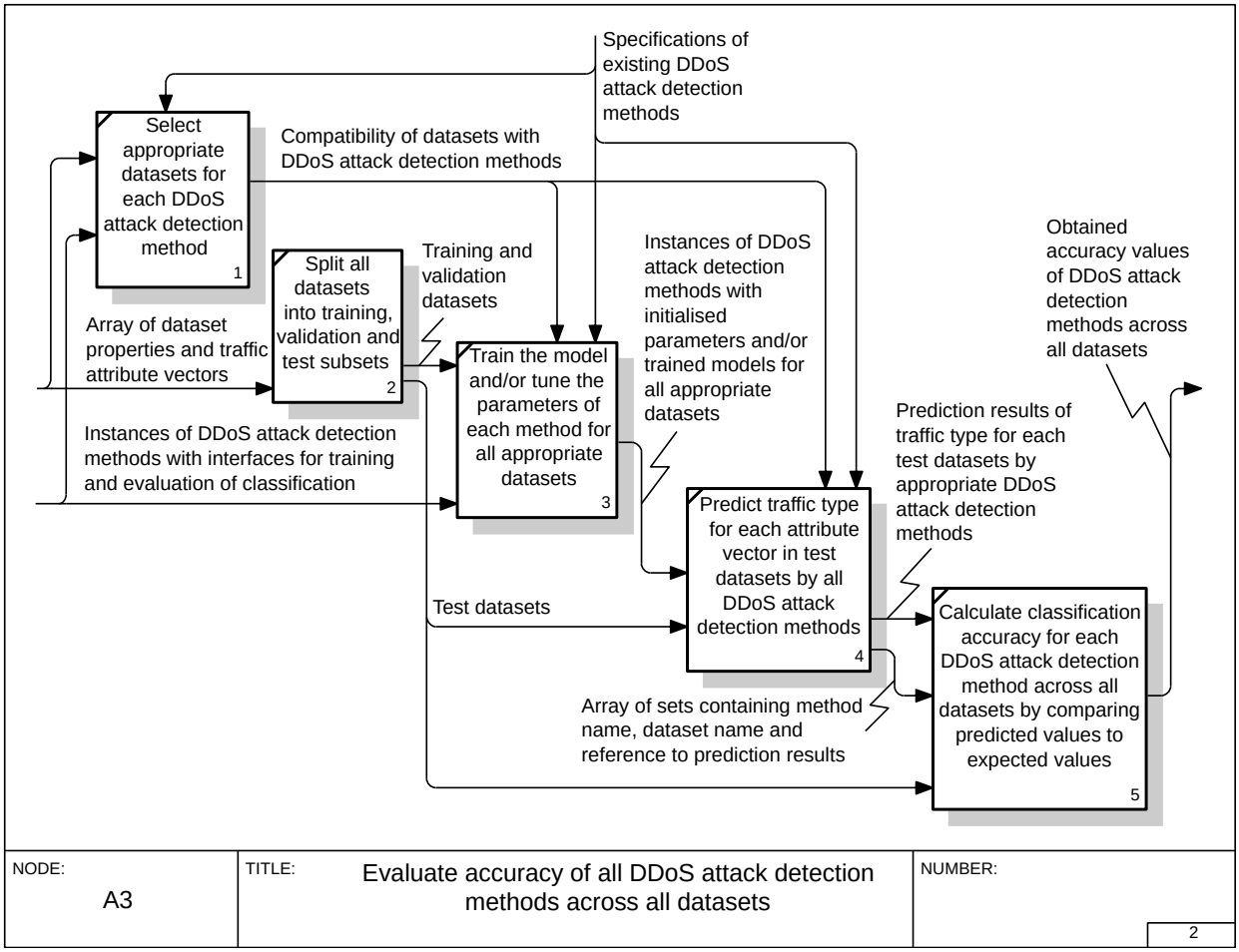

Fig. 4. IDEF0 diagram of the accuracy evaluation process for all DDoS attack detection methods on all datasets

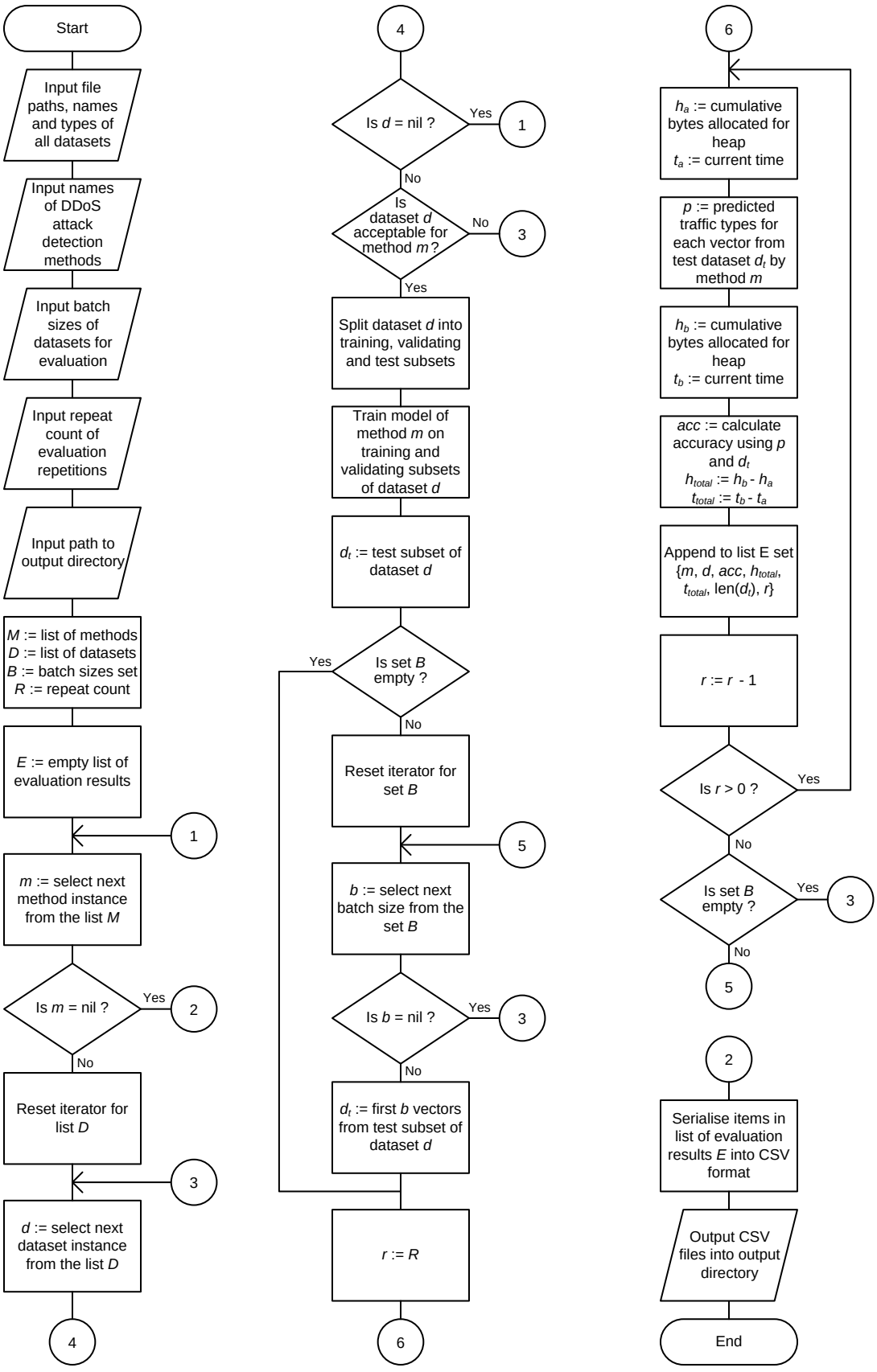

Fig. 5. Scheme of the algorithm for evaluating accuracy, operating time, memory usage of DDoS detection methods

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

The algorithm, which is illustrated in Fig. 5, outlines the steps involved in selecting datasets, training models, classifying test data, and calculating performance metrics. It accepts various input parameters, such as dataset specifications, method names, batch sizes, and repetition count. The flow of the algorithm begins by iterating through all considered DDoS attack detection methods. For each method, the algorithm processes all datasets and splits them into training and testing subsets. Once a method is trained, the algorithm determines whether batch sizes have been specified and, if so, iterates through them, slicing the test dataset accordingly. During evaluation, it measures the cumulative allocated heap bytes before and after the prediction process, as well as the time required to make predictions on the test dataset's attribute vectors. The results, including the calculated accuracy, operating time, memory usage, and associated input parameters, are then appended to a dedicated array. If a repetition count is specified, the evaluation is repeated with the same parameters for the number of provided repetitions. Finally, the consolidated results are written as CSV files. Note that the inputs of the batch size and repetition count are optional. When only the accuracy needs to be measured, the batch sizes are provided as an empty set, and the repetition count is set to zero. However, to evaluate the operating time and memory usage, the algorithm takes a set of batch sizes and a repetition count, which enables the assessment of performance across different input sizes and the generation of average results through repeated evaluations.

It is essential to examine the details of the software implementation of the proposed system, which is designed to evaluate the efficiency of DDoS attack classification methods. The source code is organized into several packages, each containing elements intended for the same purpose. The "Dataset" structure in the package of the same name represents a dataset and includes fields where relevant information is stored. Additionally, this structure implements methods for loading data from a CSV file and retrieving various data, including an array of attribute value vectors, a set of class names found in the dataset, the OSI model layer corresponding to the stored attributes, and the total number of vectors in the dataset. In the "results" package, the "PredictionResult" and "ResourceUsage" structures represent information about classification accuracy and computing resources used, respectively. Together, they are included in the "EvaluationResult" structure, which, through composition, describes the performance indicators of a method during testing on a specific dataset, with the dataset's name also reflected in the structure. Each DDoS attack detection method is evaluated through a series of experiments, during which the datasets and size of the test samples are varied. The quantitative indicators calculated during each experiment were stored in the "EvaluationResults" structure, which consisted of the method name and an array of "EvaluationResult" instances that reflected the results of each experiment. In the "methods" package, each DDoS attack detection method is encapsulated within its own structure, which implements the "DDoSDetectionMethod" interface. This interface includes several functions essential for utilizing classification methods, such as "GetName", which returns the full name of the method; "IsDatasetAccepted", used to verify compatibility of the method with a given dataset; "Train", which takes a dataset to either train the model or initialize method parameters; and "Evaluate", which assesses the method's classification accuracy using a specified test sample size and returns results. The above elements from the "methods" package implement the strategy design pattern, where the "DDoSDetectionMethodsAssessor" structure from the "benchmarks" package plays the role of the context. This structure evaluates all DDoS attack detection methods in the "EvaluateAllMethods" function, where it sequentially changes strategies, i.e., instances of the corresponding method structures, that are added in advance by the "AddDetectionMethod" function. Then, this structure executes the "Train" and "Evaluate" functions for each new strategy. In addition, when "EvaluateAllMethods" is executed, each strategy is tested on all datasets added by the "AddDataset" function. Finally, the obtained results are exported to CSV files using the "ExportEvaluationResultsToCSV" function from the "DDoSDetectionMethodsAssessor" structure. The UML class diagram for the developed structures and interfaces (Fig. 6) provides a comprehensive overview of the software system, highlighting the interrelationships among the various components.

During the implementation of the software system, third-party modules were used: sklearn [29] to search for k-nearest neighbors and construct a multilayer perceptron, go-deep [30] to generate a deep neural network model, and gonum [31] to manipulate the matrix. The experimental evaluation results of the average classification accuracy of application-layer DDoS attack detection methods [12 – 16] for all considered datasets are presented in Table 5.

As illustrated in Table 5, the DNN [16] method was the sole approach evaluated on a large-scale dataset that incorporated the generated data from all DDoS utilities [19 – 24], as it is the only method capable of performing multi-class classification. It is important to note that in this case, the average classification accuracy was calculated based on the proportion of correctly matched predictions to the expected class names for each network flow relative to the total number of vectors in the sample.

To evaluate the operating speed of the methods [12 – 16], three consecutive measurements of the classification time were taken for samples from the Benign and Hulk [23] dataset of different sizes: 100, 500, 1000, 2500, and 5000 vectors. Subsequently, the arithmetic means of the obtained values from the three measurements were calculated for each experiment. The results of measuring the operating times of the methods are given in Table 6.

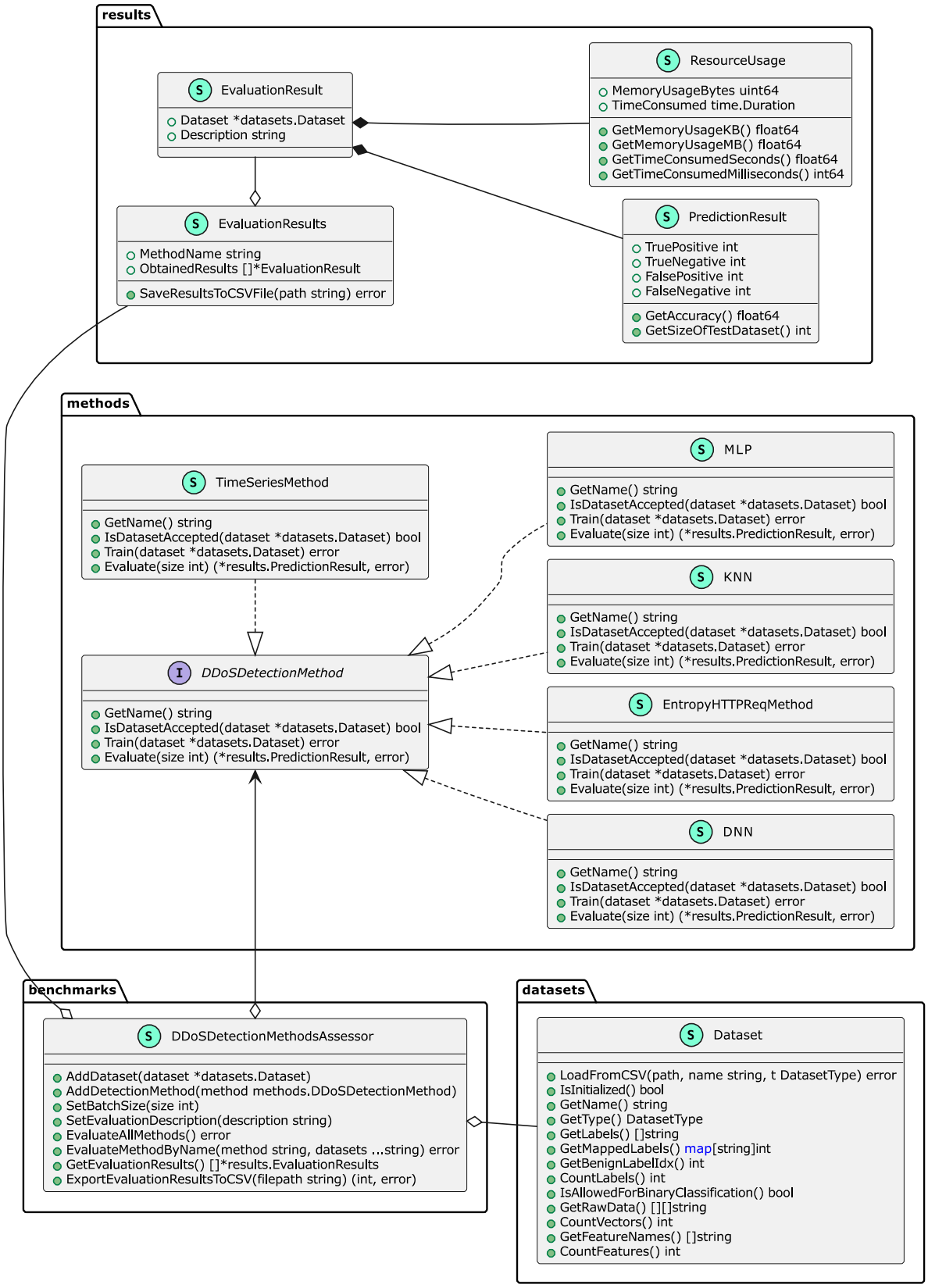

Fig. 6. UML class diagram of a software system to test DDoS attack detection methods

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Table 5

Average classification accuracy of application-layer DDoS attack detection methods on different datasets

Dataset name	Average classification accuracy of the method, %				
	EUPI-EIPU [12]	Time series [13]	KNN [14]	MLP-GA [15]	DNN [16]
Benign and Rudy [19]	91.67	94.42	99.42	97.17	99.44
Benign and Slowloris [20]	95.74	97.24	99.83	96.33	99.5
Benign and SlowHTTPTest [21]	94.55	87.04	97.91	97.01	99.07
Benign and PyDDoZ [22]	93.72	80.66	99.46	96	99.72
Benign and Hulk [23]	98.39	83.18	97.26	85.53	99.97
Benign and GoldenEye [24]	98.6	90.1	96.93	97.49	99.63
Benign, Rudy [19], Slowloris [20], SlowHTTPTest [21], PyDDoZ [22], Hulk [23] and GoldenEye [24]	–	–	–	–	98.19

Table 6

Operating time of application-layer DDoS attack detection methods for different sizes of input data

Number of vectors in the test dataset	Time consumed by the method for vector classification, ms				
	EUPI-EIPU [12]	Time series [13]	KNN [14]	MLP-GA [15]	DNN [16]
100	0.27	0.04	24.33	0.05	8
500	2	0.13	119.33	0.2	31.33
1000	4	0.24	221.33	3	64.33
2500	224.33	0.74	562.67	25.33	145.67
5000	585.33	1.33	1097	93.33	273.67

Similarly, the amount of RAM consumed by the application-layer DDoS attack detection methods was measured in each experiment, and the results are presented in Table 7.

Table 7

Memory consumption of application-layer DDoS attack detection methods for different sizes of input data

Number of vectors in the test dataset	The amount of RAM consumed by the method for vector classification, KB				
	EUPI-EIPU [12]	Time series [13]	KNN [14]	MLP-GA [15]	DNN [16]
100	436.68	17.71	4658.43	33.93	0.82
500	4832.54	123.57	23246.17	206.24	3.94
1000	10385.1	251.95	46540.19	2130.96	7.85
2500	274278.01	524.37	116123.43	14400.84	19.56
5000	723350.22	1049.29	228863.96	56322.25	39.09

In conclusion, the values of average classification accuracy (Table 5), operating time (Table 6), and memory consumption (Table 7) for the analysed DDoS attack detection methods [12 – 16] were obtained.

5. Results discussion

A brief review of the obtained results indicates that among the methods [12 – 16], no single method simultaneously achieved the highest average classification accuracy, the lowest operating time, and the lowest memory consumption. Therefore, to gain a more detailed understanding of the data presented in Tables 5–7, it is necessary to analyze each method separately, highlighting the specific advantages and limitations of each method against different DDoS utilities.

It is important to acknowledge that the accuracy of DDoS attack detection methods was evaluated using manually generated datasets, each comprising a single instance of a specific DDoS attack type generated by the corresponding DDoS utility. Furthermore, the benign traffic in each dataset was replicated and distributed evenly throughout the attack duration using the SMOTE algorithm to achieve a more balanced representation. These two factors resulted in a discrepancy between the reported average classification accuracy in this study and the accuracy that could be observed in real-world scenarios. This is due to a significant limitation of the current methodology, namely that it does not consider

dynamic network environments, where traffic characteristics can fluctuate more unpredictably. Therefore, the actual accuracy under real-life conditions may be lower than that observed in this study. Despite these limitations, the results provide a valuable basis for comparing the relative performance of the considered methods, particularly in terms of their average classification accuracy in relatively stable traffic environments. Further research can address this issue by evaluating the detection accuracy in more complex and dynamic network scenarios and employing a more comprehensive approach, thereby providing a more accurate assessment of real-life performance.

It should also be mentioned that all the experiments and measurements conducted in this research were performed on a computer with the following configuration: Intel Core i5-10210U processor with a clock speed 2.11 GHz; DDR4 RAM of 16 GB.

The EUPI-EIPU [12] method demonstrated satisfactory classification accuracy (exceeding 90%) for malicious traffic across all considered DDoS utilities. However, using this criterion, it is inferior to the majority of the other methods. For high-rate DDoS attacks (Hulk [23] and GoldenEye [24]), classification accuracy reaches 98%, while for low-rate attacks it fluctuates between 91% and 95%. This discrepancy can be explained by the lower frequency of requests during low-rate attacks, which causes the entropy value to potentially align with that of benign traffic. In addition, the entropy value of a particular URL resource is significantly influenced by the size of the time window. This parameter determines the time interval within which the collected requests are analysed and classified, segmenting the continuous stream of Internet traffic into discrete intervals, i.e., information arrays, for further processing. The findings indicated that a 30 s time window was optimal for detecting both HR- and LR-DDoS attacks, achieving the highest classification accuracy for both types compared to other time window values. The entropy threshold for this method was calculated as the average between the mathematical expectations of the entropy values for benign and malicious traffic, adjusted by subtracting and adding the standard deviation, respectively. Although the EUPI-EIPU [12] method can perform classification at high speed, this speed significantly decreases as the number of input data increases. For example, classification time increased to 0.2 s for 2500 vectors, whereas for 1000 vectors it took 0.004 s. Another disadvantage of this method is the high RAM consumption (700 MB for 5000 vectors), which is one of the highest among the evaluated methods. This high memory usage is likely due to the dynamic creation of hash tables to group requests by URL and senders by IP address during method execution, which consumes additional memory.

The highest DDoS attack classification speed was observed with the Time series [13] method, which represents a significant advantage given the urgency of reducing the time required to detect and block DDoS attacks. The Time series [13] method consumed significantly less time for classification compared to its nearest competitor, taking only 0.001 s to process 5000 vectors, whereas the MLP-GA [15] method took 0.09 s for the same task. The rationale behind this high processing speed is that the Time series [13] method does not attempt to identify attacks by processing all input data. Instead, it selects a subset of the first packets from each source (50% of the packets in this study) and predicts future delays between packets. In addition, in contrast to the other methods analysed, this method employs only a single network-layer attribute, thereby reducing the computational complexity. However, the average classification accuracy of the Time series [13] method was the lowest among the obtained results. This method relies solely on the time delay between packets, resulting in an approximate accuracy of 83% for high-rate DDoS attacks and a peak accuracy of 90% for GoldenEye [24]. In addition, the classification accuracy for HR-DDoS attacks could have been even lower because the original version was designed to detect only slow attacks, which are identified when the time delay between packets exceeds a given threshold. Consequently, the Time series [13] method was modified in this study to also detect high-rate attacks. DDoS class assignment for a client occurs when the time delay between received packets is either large or small, i.e., when it exceeds or falls below the predefined thresholds, depending on the type of attack identified during training. The classification accuracy of LR-DDoS attacks ranges from 87% to 97%, except for PyDDoZ [22], for which the generated malicious traffic was identified with the lowest accuracy of 80%. This poor accuracy is likely due to the rate of sending requests by PyDDoZ [22] being substantially similar to the rate of benign traffic. Another advantage of the Time series [13] method is that it has one of the smallest memory consumptions among the others, requiring only 1 MB for 5000 vectors.

The accuracy of the KNN [14] method fluctuates between 97% and 99% for all considered sources of DDoS attacks, making it one of the best-performing methods analysed in this study. In experiments, we determined that the optimal value for parameter k, at which classification accuracy reaches its maximum value, is 3. This corresponds to selecting the three nearest neighbours to determine the object's class. However, the KNN [14] method had the longest operating time among all considered methods, taking 1.1 s to process 5000 vectors, which is twice as slow as EUPI-EIPU [12] with the same input data. The classification process could be even longer without the use of the k-d tree (KD-

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Tree) [32] in the software implementation, which speeds up the search for nearest neighbouring objects. The low classification speed can be attributed to the use of four network-layer attributes, which significantly increases the number of computational operations while ensuring high accuracy of DDoS attack identification. Another limitation of the KNN [14] method is its high RAM consumption, which renders it one of the least memory-efficient methods to be evaluated. Notably, classifying only 5000 vectors requires 220 MB of memory. This large memory footprint can be attributed to the inherent characteristics of the KD-Tree data structure employed by KNN.

As previously stated, the MLP-GA [15] method exhibits relatively low time consumption for data classification (0.09 seconds for 5000 vectors) and moderate memory consumption, requiring only 55 MB to process 5000 vectors. It is worth noting that the sigmoid function was selected as the activation function, and Adam was employed as the optimization algorithm in the implementation of this neural network. The maximum number of training iterations in this model was 300 epochs, and an early stopping mechanism was employed to prevent overtraining. The requests submitted by each website visitor were divided into groups of 10 requests each, within which the required attributes for this method were calculated, including the entropy value of the user requests. The MLP-GA [15] achieves high average classification accuracy of 97%, with a slight reduction to 85% for requests originating from the Hulk [23] utility. During the implementation of the proposed method, it was determined that filtering web requests using the GET method, as previously assumed in the study [15], was not necessary. Therefore, the obtained accuracy values are satisfactory, particularly for low-rate attacks.

In this study, the DNN [16] method demonstrated the highest efficiency in classifying application-layer DDoS attacks, achieving an accuracy of 99% for all considered malicious traffic sources. Furthermore, the accuracy of the multi-class classification was 98%. This high performance can be attributed to the use of numerous attributes (80 in total) in detecting malicious traffic with a deep neural network, which is also known for its high performance. Although requiring a relatively low number of training iterations (10 epochs for binary classification and 30 for multi-class), this method exhibits a significantly longer training time compared to MLP-GA [15], which is one of its notable disadvantages. In addition, the operating time of DNN [16] is relatively high, taking 0.3 seconds to process 5000 vectors, which is approximately three times slower than MLP-GA [15]. However, in terms of RAM consumption, this method is the most efficient, as it uses only 0.04 MB for 5000 input vectors. This memory efficiency can be explained by the lack of need for preprocessing input data prior to classification. Furthermore, prediction operations (input data processing) in a deep neural network are executed without creating additional data structures.

Thus, among the analysed methods for detecting application-layer DDoS attacks, none simultaneously achieved the best average classification accuracy and operating time, which highlights the relevance of developing an improved method. Specifically, the average accuracy of methods with relatively short operating times (e.g., Time series [13]) can be enhanced by incorporating additional attributes from web requests. However, using an excessive number of attributes may increase the operating time of the method, as observed in DNN [16]. Therefore, it is critical to identify Internet traffic features that are highly correlated with DDoS attack requests. For example, prior to training a DNN [16] model, techniques, such as principal component analysis (PCA) or autoencoders, can be employed to reduce the dimensionality of the input data by discarding irrelevant information, which can increase the speed of DDoS classification.

6. Hypotheses and approach for improving AL-DDoS attack detection methods

Several hypotheses need to be considered to develop an application-layer DDoS attack detection method that can outperform existing methods in terms of both classification accuracy and speed.

The first hypothesis proposes improving the Time series [13] method by incorporating the analysis of information entropy changes in HTTP request attributes, in addition to examining the time delay between network packets. Specifically, further investigation can be conducted to determine whether the classification accuracy of the proposed method can be enhanced by including parameters such as the HTTP request method, request URL, number and length of request headers, and size of the request body. This approach involves constructing additional time series for these attributes and predicting their future entropy values to detect abnormal deviations from typical benign traffic values. Furthermore, parameters related to the web server, such as CPU load, RAM usage, and disk activity, could also be considered, as high values may indicate a DDoS attack. However, it must be experimentally proven that these new attributes improve the classification accuracy of the proposed method.

The second hypothesis proposes the creation of a new method to detect AL-DDoS attacks, focusing on analysing the graph representing transitions between URL resources (pages) of a website by each user to identify anomalous behavior. It is evident that the majority of ordinary users visit popular pages on a website, and the order in which they access such pages

may be slightly similar (even for different users), as they typically navigate to the next page by clicking on one of the available hyperlinks on the current page, depending on the navigation options offered on the website. In contrast, DDoS attacks are typically directed either at a specific URL or by crawling all available pages of a website. Therefore, to identify suspicious behaviour typical of DDoS attacks, it is advisable to focus on factors, such as the number of cycles, path length in the graph, and other relevant features during the analysis of the transition graph.

The main challenge in implementing the proposed hypotheses is the difficulty in capturing HTTP request attributes when the HTTPS is used, which is currently the most prevalent protocol among web servers [33]. This problem arises because all data transmitted via HTTPS are protected by asymmetric cryptography, which makes it impossible to decrypt the value of any web request attribute without a private key. It is obvious that providing access to private keys to third-party software is a risky and unsafe practice. Therefore, a more reliable alternative approach is preferred. One potential solution is to use a specific software architecture for designing Internet services, which enables the encryption of requests not directly within the web framework (i.e. the program that directly processes web requests and is responsible for the functioning of the website) but in a separate module for traffic routing. This module allows the user to inspect web requests in unencrypted form for software extensions within the system. For example, Istio, a framework created for Kubernetes clusters to configure networks for microservices and control the routing of Internet traffic between clusters, can serve as such a router. Istio allows for the encryption of web requests sent outside the internal system to the global network, the setting of special rules, and the integration of middleware to process HTTP requests, thereby providing access to all attributes of the requests [34]. In other words, it is possible to read all parameters of web requests in Istio, even when using the HTTPS protocol by implementing an application embedded in Istio that uses the provided interface for this purpose. Furthermore, the Istio framework was adapted for the Kubernetes platform, which is the most popular environment for deploying microservices [35]. Istio is widely used in cloud services, including Kubernetes-based projects. Therefore, an application-layer DDoS attack detection software system is proposed exactly for the Kubernetes platform, where the attributes of Internet traffic are collected using the Istio framework.

Based on the above hypotheses and software architecture, a potential approach to improving current application-layer DDoS attack detection methods is described below, with a focus on software engineering aspects. As previously stated, DDoS detection software should be deployed on a Kubernetes cluster and interact with the Istio framework. Given the increasing popularity of microservice architecture and containerisation (a method of virtualizing and packaging applications and their dependencies in an isolated container that allows them to run in any environment), this requirement is reasonable. Kubernetes is designed for applications that require high performance, fault tolerance, and scalability. Consequently, multiple websites can be hosted on the same cluster, thereby allowing the DDoS detection system to effectively monitor and protect all deployed web servers by leveraging the cluster's distributed computing resources. This approach assumes that traffic encryption for HTTPS (TLS termination) is performed in a virtual gateway in Istio that connects the cluster's internal network to the Internet. To capture the unencrypted web request attributes required for the proposed DDoS detection methods, an EnvoyFilter [36] should be created in the Istio network, through which all Internet traffic directed to the web servers will pass. This filter forwards each HTTP request to middleware, where the required attributes of the web requests are sent to a separate software module for further classification of DDoS attacks. Communication between different software components in the system should be performed using a message broker. This facilitates the decoupling of components, their asynchronous interactions, scalability, and fault tolerance. The web request classification module should implement one of the proposed DDoS detection methods and be able to scale horizontally during peak loads. Identified sources of malicious traffic can be stored in a non-relational key-value database, where it is sufficient to retain information about the IP address of each DDoS generator and the time of its detection. These data can be further used to create rules to block Internet traffic from DDoS sources in the Istio network.

Conclusions

This study analyses modern methods to detect application-layer DDoS attacks on web server resources. This study examines several relevant methods in this field, revealing their main concepts, operational mechanisms, advantages, and disadvantages. Furthermore, this study evaluates the average classification accuracy of these methods and measures their operating time using generated datasets that include examples of malicious requests from popular utilities to conduct low-rate and high-rate DDoS attacks at the HTTP protocol level. The experimental results demonstrate that none of the analysed methods achieves high average classification accuracy and operating speed simultaneously, which indicates the need to develop a new method to detect application-layer DDoS attacks

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

with the desired characteristics. Two hypotheses are proposed for creating new methods, one of which involves the additional use of HTTP request attributes in the classification of Internet traffic, and the other involves an analysis of user transition graphs between website pages. In addition, this paper presents a brief overview of a potential architectural solution within the proposed approach for a software system designed to detect DDoS attacks based on the Kubernetes platform and the Istio framework. If further developed and implemented, this solution could address the challenge of collecting web request parameter values for websites that use the cryptographically protected HTTPS protocol.

The scientific novelty of this study lies in obtaining average classification accuracy values, as well as measuring the operating time and RAM consumption during the classification of network packets and web requests from prepared datasets under the same conditions for six modern and promising methods for detecting application-layer DDoS attacks, namely: EUPI-EIPU [12], Time series [13], KNN [14], MLP-GA [15], and DNN [16]. To conduct this experimental study, balanced datasets were generated with malicious requests created using popular DDoS utilities: Rudy [19], Slowloris [20], SlowHTTPTest [21], PyDDoZ [22], Hulk [23], and GoldenEye [24]. The results demonstrate that the Time series [13] method exhibits the shortest operating time (1.33 ms for 5000 vectors), and the DNN [16] method exhibits the highest average classification accuracy (ranging from 99.07% to 99.97%) and the lowest memory consumption (39.09 KB for 5000 vectors). In addition to the comparative analysis of existing DDoS attack detection methods based on the obtained values, the scientific novelty lies in formulating two hypotheses to create a new method to identify DDoS attacks at the HTTP protocol level, which will combine high accuracy and classification speed to outperform existing analogues in these metrics.

The practical significance of this study lies in the development of a software system to test DDoS attack detection methods, as well as in the software implementation of all analysed methods. In future, this system will enable the evaluation of new methods by measuring the accuracy, operating time, and memory usage. In addition, the practical significance is demonstrated by describing a potential software architecture for a system that collects web request attributes using a dedicated software module for Istio in a Kubernetes cluster. Although the architectural design is presented in a relatively superficial way, it nevertheless provides a conceptual framework that can be further developed and refined in future research.

A potential approach to improve existing methods for detecting application-layer DDoS attacks is proposed. This approach encompasses the realization of the formulated hypotheses to improve DDoS attack classification methods, along with the implementation of the proposed software system architecture. A concise description is provided that offers insights into the processes involved in collecting, processing, and classifying Internet traffic data from a software engineering perspective.

Future research directions. Future research will focus on the development and validation of the hypotheses proposed in this paper to develop a method for detecting application-layer DDoS attacks with the highest accuracy and classification speed among known analogues. This involves software implementation of the methods based on the proposed hypotheses and evaluation using the developed information technology. Furthermore, the proposed approach requires further refinement and implementation of the outlined architectural design, and the resulting software system should then be experimentally evaluated in the Kubernetes cluster environment.

In addition, the next steps involve developing a new methodology to evaluate the accuracy of DDoS attack detection methods. This methodology will be sensitive to sudden and significant traffic changes in dynamic network environments, which will make it more applicable to real-world scenarios. The current approach is insufficient to address the issue of traffic fluctuations and non-stationary behaviours inherent to DDoS attacks, which limits the reliability of the obtained results. The implementation of a more comprehensive evaluation approach will facilitate precise measurement of the classification accuracy of DDoS attack detection methods. Furthermore, the methodology for generating and balancing DDoS datasets requires further revision. In particular, the use of the SMOTE algorithm to balance datasets introduces a limitation because it generates synthetic data without considering the non-stationary nature of DDoS attacks. Therefore, it is necessary to employ a more sophisticated algorithm to address the imbalances in the datasets, ensuring that the generated data more accurately suit the dynamic network environment. Moreover, future evaluations may consider testing detection methods on datasets containing multiple instances of different types of DDoS attacks potentially generated by different DDoS utilities to simulate more complex and realistic attack scenarios. The implementation of these improvements will lead to more precise assessments of the accuracy of DDoS attack detection methods, which will enhance the reliability of experimental results.

Contributions of authors: conceptualisation, problem statement, literature review, analysis of methods for detecting application-layer DDoS attacks, method-

ology, software implementation of methods and development of a system for evaluating them, experimental research, analysis of the obtained results, formulation of the novel approach, hypotheses, and conclusions – **Arkadii Kravchuk**; review, editing – **Mykola Onai**.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

This study was conducted without financial support.

Data availability

The manuscript contains no associated data.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies while creating the presented work.

All the authors have read and agreed to the publication of the finale version of this manuscript.

References

1. Simons, G., Danyk, Y., & Maliarchuk, T. Hybrid war and cyber-attacks: creating legal and operational dilemmas. *Global Change, Peace & Security*, 2020, vol. 32, no. 3, pp. 337–342. DOI: 10.1080/14781158.2020.1732899.

2. Uma, M., & Padmavathi, G. A Survey on Various Cyber Attacks and their Classification. *International Journal of Network Security*, 2013, vol. 15, no. 5, pp. 390–396.

3. Kizzee, K. *Cybersecurity: Cyber Attack Statistics to Know*. Parachute Technology. Available at: https://parachute.cloud/cyber-attack-statistics-data-and-trends/ (accessed 01.01.2024).

4. *Cyber Dimensions of the Armed Conflict in Ukraine: Quarterly Analysis Report Q3 from July to September 2023*. CyberPeace Institute. Available at: https://cyberpeaceinstitute.org/wp-content/uploads/2023/12/Cyber-Dimensions_Ukraine-Q3-2023.pdf (accessed 01.01.2024).

5. Ohsita, Y., Ata, S., & Murata, M. Detecting distributed denial-of-service attacks by analyzing TCP SYN packets statistically. *IEICE transactions on communications*, 2006, vol. 89, no. 10, pp. 2868–2877. DOI: 10.1093/ietcom/e89-b.10.2868.

6. Bogdanoski, M., Shuminoski, T., & Risteski, A. Analysis of the SYN Flood DoS Attack. *International Journal of Computer Network and Information Security*, 2013, vol. 5, no. 8, pp. 1–11. DOI: 10.5815/ijcnis.2013.08.01.

7. Boro, D., Basumatary, H., Goswami, T., & Bhattacharyya, D. K. UDP flooding attack detection using information metric measure. *Proceedings of International Conference on ICT for Sustainable Development*, 2016, vol. 408, pp. 143–153. DOI: 10.1007/978-981-10-0129-1_16.

8. *Application layer DDoS attack: an overview*. Cloudflare, Inc. Available at: https://www.cloudflare.com/learning/ddos/application-layer-ddos-attack/ (accessed 01.01.2024).

9. Mantas, G., Stakhanova, N., Gonzalez, H., Jazi, H. H., & Ghorbani, A. A. Application-layer denial of service attacks: taxonomy and survey. *International Journal of Information and Computer Security*, 2015, vol. 7, no. 2-4, pp. 216–239. DOI: 10.1504/ijics.2015.073028.

10. Kaur, P., Kumar, M., & Bhandari, A. A review of detection approaches for distributed denial of service attacks. *Systems Science & Control Engineering*, 2017, vol. 5, no. 1, pp. 301–320. DOI: 10.1080/21642583.2017.1331768.

11. No. G., & Ra, I. An efficient and reliable DDoS attack detection using a fast entropy computation method. *International Symposium on Communications and Information Technology*, 2009, pp. 1223–1228. DOI: 10.1109/iscit.2009.5341118.

12. Zhao, Y., Zhang, W., Feng, Y., & Yu, B. A classification detection algorithm based on joint entropy vector against application-layer DDoS attack. *Security and Communication Networks*, 2018, vol. 2018, article no. 9463653. 8 p. DOI: 10.1155/2018/9463653.

13. Laptyev, O. A., Buchyk, S. S., Savchenko, V. A., Nakonechnyy, V. S., Mykhal'chuk, I. I., & Shestak, Ya. V. Vyyavlennya ta blokuvannya povil'nykh DDoS-atak za dopomohoyu prohnozuvannya povedinky korystuvacha [Detecting and blocking slow DDoS attacks by predicting user behaviour]. *Naukoyemni tekhnolohiyi – Science-intensive technologies*, 2022, vol. 3, no. 55, pp. 184–192. DOI: 10.18372/2310-5461.55.16908. (In Ukrainian).

14. Dong, S., & Sarem, M. DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks. *IEEE Access*, 2020, vol. 8, pp. 5039–5048. DOI: 10.1109/access.2019.2963077.

15. Johnson Singh, K., Thongam, K., & De, T. Entropy-based application layer DDoS attack detection using artificial neural networks. *Entropy*, 2016, vol. 18, no. 10, article no. 350. 17 p. DOI: 10.3390/e18100350.

16. Muraleedharan, N., & Janet, B. A deep learning based HTTP slow DoS classification approach using

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

flow data. *ICT Express*, 2021, vol. 7, no. 2, pp. 210–214. DOI: 10.1016/j.icte.2020.08.005.

17. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. A survey of network-based intrusion detection data sets. *Computers & Security*, 2019, vol. 86, pp. 146–147. DOI: 10.1016/j.cose.2019.06.005.

18. Kumar, V., Kumar, K., & Mahadev. Classification of DDoS attack tools and its handling techniques and strategy at application layer. In *2nd International Conference on Advances in Computing, Communication, & Automation*, 2016. 6 p. DOI: 10.1109/icaccaf.2016.7749002.

19. *RUDY attack tool to perform slow-rate attacks*. GitHub, Inc. Available at: https://github.com/darkweak/rudy (accessed 01.02.2024).

20. *Slowloris HTTP denial of service attack tool in Python*. GitHub, Inc. Available at: https://github.com/gkbrk/slowloris (accessed 01.02.2024).

21. *SlowHTTPTest application layer denial of service attacks tool*. GitHub, Inc. Available at: https://github.com/shekyan/slowhttptest (accessed 01.02.2024).

22. *PyDDoZ: DDoS tool using application layer (L7) attack techniques*. GitHub, Inc. Available at: https://github.com/ProTechEx/pyddoz (accessed 01.02.2024).

23. *Hulk (Http Unbearable Load King) DDoS attacking tool*. GitHub, Inc. Available at: https://github.com/grafov/hulk (accessed 01.02.2024).

24. *GoldenEye Layer 7 DDoS test tool*. GitHub, Inc. Available at: https://github.com/jseidl/GoldenEye (accessed 01.02.2024).

25. Kravchuk, A. *Source code of website "Rapid delivery"*. GitHub, Inc. Available at: https://github.com/akrava/rapid-delivery/tree/mvc-ssr (accessed 01.02.2024).

26. *Python CICFlowMeter: CICFlowMeter Python Implementation*. GitHub, Inc. Available at: https://github.com/hieulw/cicflowmeter (accessed 01.02.2024).

27. *Pyshark: Python wrapper for tshark, allowing python packet parsing*. GitHub, Inc. Available at: https://github.com/KimiNewt/pyshark (accessed 01.02.2024).

28. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. SMOTE: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 2002, vol. 16, pp. 321–357. DOI: 10.1613/jair.953.

29. *Sklearn: partial port of scikit-learn to go*. GitHub, Inc. Available at: https://github.com/pa-m/sklearn (accessed 01.03.2024).

30. *Go-deep: neural network implementation for deep learning*. GitHub, Inc. Available at: https://github.com/patrikeh/go-deep (accessed 01.03.2024).

31. *Gonum: set of numeric libraries for the Go programming language*. GitHub, Inc. Available at: https://github.com/gonum/gonum (accessed 01.03.2024).

32. Tiwari, V. R. Developments in KD Tree and KNN Searches. *International Journal of Computer Applications*, 2023, vol. 185, no. 17, pp. 17–23. DOI: 10.5120/ijca2023922879.

33. *Usage statistics of Default protocol https for websites*. W3Techs. Available at: https://w3techs.com/technologies/details/ce-httpsdefault (accessed 01.03.2024).

34. *Current State and Future of the Istio Service Mesh*. Tetrate. Available at: https://7637559.fs1.hubspotusercontent-na1.net/hubfs/7637559/Istio%20Book/The-Current-State-and-Future-of-the-Istio-Service-Mesh.pdf (accessed 01.03.2024).

35. Shurupov, D. *Kubernetes and containerization trends according to reports of 2021*. Palark GmbH. Available at: https://blog.palark.com/kubernetes-and-containers-market-trends-2021/ (accessed 01.03.2024).

36. Toader, S. *How to write WASM filters for Envoy and deploy it with Istio*. Outshift by Cisco Systems, Inc. Available at: https://outshift.cisco.com/blog/envoy-wasm-filter (accessed 01.03.2024).

Received 27.05.2024, Accepted 20.08.2024

РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ОЦІНКИ ТА ВДОСКОНАЛЕННЯ МЕТОДІВ ВИЯВЛЕННЯ DDOS-АТАК ПРИКЛАДНОГО РІВНЯ

А. А. Кравчук, М. В. Онай

Предметом дослідження в статті є методи розпізнавання DDoS-атак на рівні протоколу HTTP для обґрунтування вимог до створення програмного забезпечення для ідентифікації зловмисних клієнтів вебсерверів. **Метою** є розроблення інформаційної технології для оцінки ефективності методів виявлення DDoS-атак, яка дозволить кількісно визначити час їх роботи, споживання пам'яті та приблизну точність класифікації. Крім того, ця стаття має на меті запропонувати гіпотези та перспективний підхід для вдосконалення наявних методів виявлення DDoS-атак прикладного рівня з метою збільшення їхньої точності та швидкості ідентифікації. **Завдання**: проаналізувати сучасні методи виявлення DDoS-атак прикладного рівня, дослідити їх особливості та недоліки; розробити програмний комплекс для оцінки методів виявлення DDoS-атак; програмно

реалізувати розглянуті методи і експериментально виміряти показники їх роботи, а саме: точність класифікації, час роботи, використання пам'яті; порівняти ефективність досліджених методів; сформулювати гіпотези та підхід для вдосконалення наявних та/або розроблення нових методів на основі отриманих результатів. **Методами**, що використовуються, є: абстрагування, аналіз, системний підхід, емпіричне дослідження. Зокрема, здійснено збір даних Інтернет-трафіку, що генерувався утилітами для проведення DDoS-атак, та виконано оброблення цих наборів даних за допомогою методу SMOTE (synthetic minority over-sampling technique) для їх збалансування. Крім цього, програмно реалізовано методи, що досліджуються, включно з підбором необхідних параметрів та навчанням моделей штучних нейронних мереж, для їх оцінки та аналітичного порівняння. Отримано такі **результати**. Обчислено значення показників середньої точності класифікації, а також часу роботи та обсягів споживання оперативної пам'яті під час виконання класифікації Інтернет-трафіку для шістьох методів виявлення DDoS-атак за однакових умов. Доведено, що розроблення нового методу виявлення DDoS-атак на рівні протоколу HTTP із кращими значеннями показників точності та часу класифікації є актуальним завданням. Результати експериментів показали, що метод на основі аналізу часових рядів має найменший час роботи (1,33 мс для 5000 векторів), а метод на основі глибокої нейронної мережі – найвищу середню точність класифікації (від 99,07 % до 99,97 %) та найменші обсяги використання оперативної пам'яті (39,09 КБ для 5000 векторів). **Висновки**. У даному дослідженні розроблено програмну систему для оцінки середньої точності класифікації методів виявлення DDoS-атак та вимірювання обчислювальних ресурсів, які вони використовують. Наукова новизна отриманих результатів полягає в формулюванні двох гіпотез та перспективного підходу щодо створення нового методу ідентифікації DDoS-атак на рівні протоколу HTTP, який матиме одночасно високу точність класифікації та швидкість роботи для того, щоб перевершити досліджені аналоги за цими показниками. Зокрема, одна з гіпотез ґрунтується на додатковому застосуванні атрибутів HTTP-запитів під час класифікації Інтернет-трафіку, а інша має здійснювати аналіз графа переходів користувачів між сторінками вебсайту. Також у статті поверхнево описано перспективний підхід, який полягає в реалізації описаних гіпотез та впровадженні запропонованої архітектури програмної системи виявлення DDoS-атак прикладного рівня для платформи Kubernetes і фреймворку Istio, що вирішує питання збору значень параметрів вебзапитів для сайтів, які використовують криптографічно захищений протокол HTTPS.

Ключові слова: DDoS; виявлення DDoS-атак; аналіз мережевого трафіку; захист інформації; AL-DDoS; HTTP; криптографія; програмна система; Kubernetes; Istio.

Кравчук Аркадій Андрійович – асп. каф. програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Онай Микола Володимирович – канд. техн. наук, доц., доц. каф. програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Arkadii Kravchuk – PhD Student of the Department of Computer Systems Software, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine,
e-mail: arkakrava@gmail.com, ORCID: 0000-0002-6128-206X.

Mykola Onai – PhD, Associate Professor, Associate Professor at the Department of Computer Systems Software, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine,
e-mail: onay@pzks.fpm.kpi.ua, ORCID: 0000-0002-4938-8355, Scopus Author ID: 57204924611.