K-means Clustering of Honeynet Data with Unsupervised Representation Learning

Antonina Kashtalian[a], Tomas Sochor[b]

[a] *Khmelnitsky National University, Khmelnitsky, Ukraine*
[b] *Prigo University, Havirov, Czech Republic*

Abstract

Networks connected to the Internet are vulnerable to malicious activity that threaten the stability of work. The types and characteristics of malicious actions are constantly changing, which significantly complicates the fight against them. Attacks on computer networks are subject to constant updates and modifications. Modern intrusion detection systems should ensure the detection of both existing types of attacks and new types of attacks about which there might be no information available at the time of attack. Honeypots and honeynets play an important role in monitoring malicious activities and detecting new types of attacks. The use of honeypots and honeynets has significant advantages: they can protect working services, provide network vulnerability detection, reduce the false positive rate, slow down the influence of malicious actions on the working network, and collect data on malicious activity. The analysis of the data collected by a honeynet helps detect attack patterns that can be used in intrusion detection systems. This paper uses clustering to determine attack patterns based on the time series of attacker activity. Using time series instead of static data facilitates the detection of attacks at their onset. This paper proposes the joint application of k-means clustering and a recurrent autoencoder for time series preprocessing. The weights of the recurrent autoencoder are optimized on the basis of the total loss function, which contains two components: a recovery loss component and a clustering loss component. The recurrent encoder, consisting of convolutional and recurrent blocks, provides an effective time series representation, suitable for finding similar patterns using k-means clustering. Experimental research shows that the proposed approach clusters malicious actions monitored by a honeynet and identifies patterns of attacks.

Keywords

Honeynet, attack patterns, time series, k-means clustering, deep recurrent autoencoder

1. Introduction

Detecting malicious activity is critical for Internet-connected computer networks. Malicious actions that occur on the network threaten the stability of its operation and the security of confidential information available on the network. Malicious actions include a wide range of activities such as attempts to destabilize the computer network, attempts to gain access to user accounts and files, and attempts to interfere with software.

Detecting attacks on computer networks is challenging because these attacks are not static, that is, their characteristics change in time and space, making them difficult to recognize. Therefore, it is difficult to create a list of possible network attacks in advance, considering the ever-present appearance of both new attacks and modifications of existing attacks. Thus, intrusion detection systems must be able to detect various types of attacks, taking into account those about which no prior

IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine
EMAIL: yantonina@ukr.net (A. Kashtalian); tomas.sochor@osu.cz (T. Sochor)
ORCID: 0000-0002-4925-9713 (A. Kashtalian); 0000-0002-1704-1883 (T. Sochor)


information exists. The approaches used to detect network attacks can be divided into three base groups: signature-based methods, supervised learning methods, and unsupervised methods.

Signature-based attack detection methods involve first extracting certain features from the network data and then comparing with a set of pre-installed signatures. Signature-based methods are effective for detecting known attacks but are completely unsuitable for detecting new types of attacks and modifications of existing attacks. At the same time, building new signatures is costly.

Supervised learning methods require labeled datasets, on the basis of which models are trained and then used to detect malicious actions. Supervised models are often trained to detect network anomalies, so they are also suitable for identifying new attacks that have characteristics similar to existing signatures. In addition, supervised models make hyper-parameter optimization, or tuning, possible. Like signature-based intrusion detection tools, supervised models require previous costs; that is, marked datasets are necessary for training.

Unsupervised methods, unlike signature-based and supervised learning methods, require only data collection and not the existence of previously known costs. Unlabeled data is used to train unsupervised models. In many cases, unsupervised anomaly detection is used as the unsupervised method. One approach to this anomaly detection is to build models that describe the normal operation of a network, with traffic in which there are no intrusions. Then any deviations from normal behavior can be considered as an anomaly and a possible malicious action. Such models require datasets of normal data for training. Often, instead of labeled datasets, the intrusion detection system works with large unlabeled data arrays that contain both normal traffic data and a certain proportion of malicious traffic. The analysis of such large data arrays is quite challenging.

To collect data corresponding only to malicious traffic, honeypots and honeynets are used. The main purpose of a honeynet is to be attacked by an intruder, diverting his attention from working services and slowing the spread of the attack, while simultaneously collecting information about the malicious actions [1]. By design, the network traffic observed on honeypots is malicious. The use of honeypots and honeynets in computer networks has a number of advantages, in particular: maintaining independence from the working environment; identifying zero-day vulnerabilities; reducing the false positive rate; and collecting data on malicious acts [2]. Collecting malicious activity data in working computer networks is problematic because it is necessary to first distinguish between malicious and normal activity. The analysis of data collected by a honeypot involves identifying attack patterns for further use in intrusion detection systems. One of the most appropriate approaches for identifying patterns is clustering. Identifying patterns is not a trivial task, as clustering methods common for all use cases do not exist. That is, any clustering approach first requires adaptation in order to detect patterns of malicious activity in honeynets. Honeypot data can also be used to analyze malicious activity dynamically, which helps solve the problem of early attack detection.

2. State of the Art

A variety of supervised and unsupervised learning methods as well as many data preprocessing and analysis methods have been actively researched and used to detect and prevent threats in computer networks. Lysenko et al. [3] researched low-speed DDoS detection based on features inherent in botnets and the analysis of network traffic self-similarity using the Hurst coefficient. Savenko et al. [4] proposed a method of generating malicious software signatures in which each signature consists of two components: (1) the call frequency; and (2) the nature of critical API call interaction. Analysis of these signatures allows us to determine the distribution of critical API calls to malicious activity groups and to distinguish between malicious and normal traffic. Artificial immune systems using a clonal selection algorithm have been proposed for botnet detection [5]. Finally, a support vector machine based self-adaptive system, able to detect cyber-attacks of botnets, detect the botnets themselves, and configure security scenarios has been proposed. This system can recognize attacks by taking into account network activity and captured network traffic [6].

The development and application of unsupervised methods, including pattern detection and clustering, in intervention detection systems is not new, as new types of intrusions make it difficult to use supervised methods, which require labeled data. Portnoy et al. [7] propose a method based on

hierarchical clustering and unsupervised anomaly detection, trained exclusively on unlabeled data, that is able to detect various types of intrusions and maintain a low false positive rate. Their algorithm initializes an empty set of clusters and then calculates the final set of clusters in one pass, so that the calculation time is linear, but therefore such an algorithm is not the most efficient. Mazel et al. [8] also offers a completely unsupervised method that does not require a previous labeling of data and is not based on the preliminary information about distribution, so it can be immediately applied to monitor network anomalies based on the detection of anomalies and the identification of clusters of small sizes. K-means clustering is used to identify the actions of malicious programs based on information about the features of these actions [9]. Mohiuddin Ahmed et al. [10] discusses the framework for detecting DoS attacks based on partitional clustering, a DoS attack is considered as a collective anomaly, which is a pattern in data when a group of similar instances has anomalous behavior compared to other data. Taheri et al. [11] proposes an incremental clustering algorithm for finding clusters of attacks of various types, the work of which is based on the use of two subsets of clusters, one of which is stable, and distances between centroids of clusters from another subset. Different clustering methods are also used when analyzing system logs, which contain information about most events that occur in the network [12]. It is performed both static clustering, when each system log entry is considered as a separate point, and dynamic clustering, which takes into account the sequences of entries and discovers more complex dependencies.

Fuzzy clustering methods are also investigated and used. Fuzzy c-means cluster analysis of functions obtained from DNS message payloads is the basis of a botnet detection method that uses DNS-based evasion methods [13]. The stability adaptive network reconfiguration is provided based on semi-supervised fuzzy c-means clustering analysis of the features of the collected Internet traffic . [14]. Fuzzy c-means clustering is used to define security scenarios [15]. Clustering objects are feature vectors which elements can indicate the appearance of threats in computer networks.

Among advanced approaches to clustering in solving cybersecurity problems, it is worth noting a cognitive clustering algorithm that is based on the use of a probabilistic neural network model [16]. The architecture of the model is multilayer with excitatory and inhibitory neurons providing direct and backward connections between layers of neurons. Due to the significant development of neural network approaches, deep learning is also used for clustering in solving many problems [17].

In general, deep clustering methods can be divided into three groups: direct deep cluster optimization methods; methods that are based on generative models; methods based on the use of autoencoders. Direct deep clustering methods involve creating a single deep model that performs feature extraction and clustering by using the clustering loss function directly to optimize the deep neural network model [18]. Variational autoencoders and generative adversarial networks (GAN) are used as generative models. In the process of training the optimization of a reconstruction loss and Kullback–Leibler divergence is performed to determine the distribution of features across clusters, which allows interpolation of new samples [19]. Among GAN solutions, semi-supervised and completely unsupervised models are offered [20], which provide insignificant calculation costs.

The autoencoders used for the clustering task have one thing in common in that they consist of an encoder unit and a decoder unit, and are effectively trained with the use of the reconstruction loss function. The encoder is designed to obtain a representation of the characteristics of the original data object. Despite the general structure of most autoencoders, their architecture differs significantly depending on the type of data for which it is designed. Autoencoders themselves are used for noise filtering tasks [21], improving signal quality in images [22], image segmentation [23], audio signal processing [24], etc. Auto encoders provide an efficient representation of the features of the input data at the output of the encoder, which can be used not only to reconstruct this data, but also for the tasks of the dimensionality reduction of data [25], classification [26] and clustering.

The methods for joint optimization of the representation of raw data features and clustering are already used in certain applications, in particular for clustering images, in bioinformatics [27]. The paper [28] describes convolutional neural network for obtaining image features, iterative clustering of features based on standard k-means clustering method and accounting of clusters for updating weight coefficients during training. In addition to the methods that work with a previously known number of clusters, the methods are also investigated that involve determining the number of clusters in the clustering process itself. Mautz et al. [29] proposes a method of separating hierarchical clustering deep embedded cluster tree, for which it is not necessary to know the number of clusters. Sohil Atul

Shah et al. [30] also investigates the clustering algorithm, which performs simultaneous reduction in the dimension of the original data and clustering. The dimensionality reduction of data is performed by the autoencoder optimized during clustering.

The task of finding patterns and clustering anomalies of network traffic is relatively widely investigated [31], while much less research is devoted to analyzing the honeynets traffic.

3. K-means Clustering with Recurrent Autoencoder

Clustering data changed over time is complicated due to the opacity of the similarity measure compared to static data. In addition, time series can have different dimensions and the relationships between them often have an implicit nature. Direct clustering of time series requires the use of high-dimensional data and is often unable to detect hidden relationships between objects. Therefore, in most cases, prior extraction of feature is required, that is, the transformation into a feature space suitable for correct clustering.

3.1. The collection of data by the honeynet

The honeynet, part of which is shown in Figure 1, is a multilevel set of intelligent honeypots that continuously monitor the network. The honeynet includes external honeypots connected to the Internet before the firewall, honeypots located in DMZ zone, internal honeypots located inside the network. The honeynet with static honeypots contain virtual ones. An important advantage of such a honeynet is the ability to track the spread of distributed attacks and detect their patterns. This process in most cases is divided into three stages [32]: a) selection and removal of features and presentation of patterns; b) determining similarity measures suitable for pattern detection; c) grouping of similar patterns.

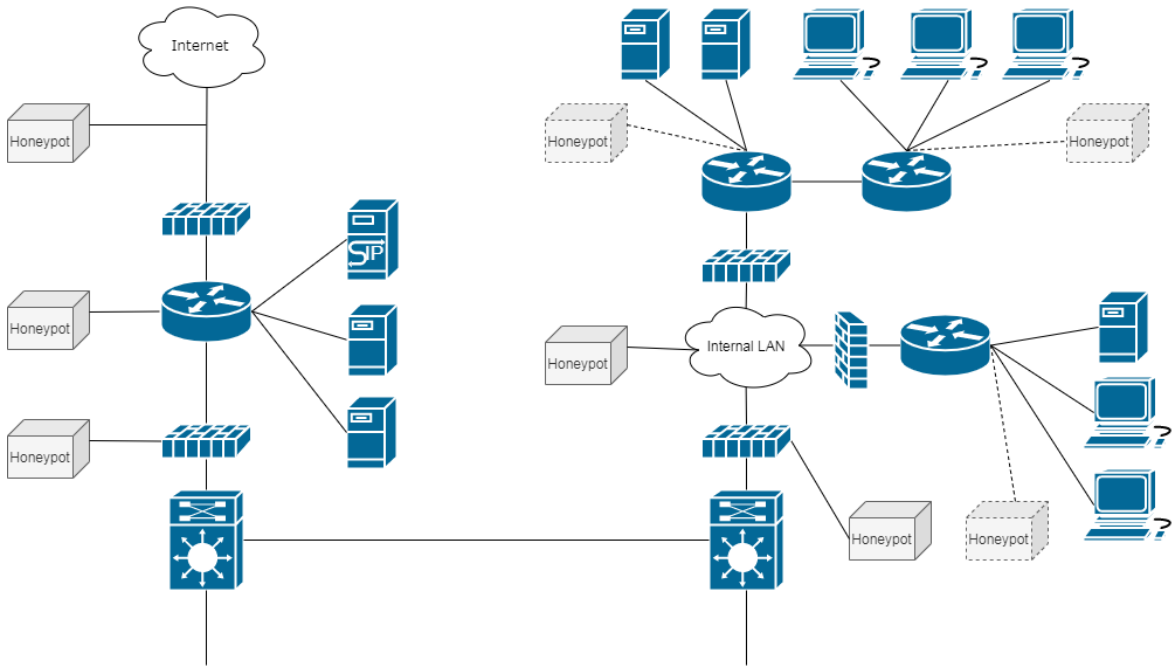

Figure 1: A Part of Honeynet

The first step involves extracting from the data array certain features that represent the relevant characteristics of malicious actions. Data collected by the honeynet and data obtained from their analysis: the number of malicious sources per time unit; the number of new malicious sources per time unit; the number of malicious sources in a particular IP aggregation; the number of malicious sources relative to the number of attacks; the number of packets received per time unit; the volume of data received per time unit; the number of packets sent per time unit; the volume of data sent per time unit; the number of e-mails received per time unit; the volume of data per message; the number of

sessions per time unit; session duration; time between sessions; “lifetime” of a malicious source. These data form time series of malicious activity on honeypots:

$$X^1 = \mathbf{x}_1^1, \mathbf{x}_2^1, \dots, \mathbf{x}_t^1, X^2 = \mathbf{x}_1^2, \mathbf{x}_2^2, \dots, \mathbf{x}_t^2, \dots X^n = \mathbf{x}_1^n, \mathbf{x}_2^n, \dots, \mathbf{x}_t^n,$$

here n – the number of honeypots in the honeynet, t – duration of the time window. The time window shifts in time with a step, the value of which depends on the time required for calculations. Feature data are numeric; the values of different features are in different ranges, so they require pre-scaling. It is expediently to use scaling in the range [0, 1] of each feature. Time series of each feature $\{x_1, x_2, \dots, x_n\}$ is converted into a series $\{x'_1, x'_2, \dots, x'_n\}$ according to the relation $x' = \frac{x - x_{min}}{x_{max} - x_{max}}$.

The data that the honeynet collects is a time series that is actually continuous over time. To cluster and identify patterns, it is necessary to select a time window within which the analysis will be conducted. To select the window duration, two approaches can be selected: 1) a time window of fixed duration t_w, which shifts in time in increments t_s; 2) a time window derived from the previous segmentation. The first approach is simple and requires only two values. Bayesian change point analysis can be used to obtain a time window based on segmentation [33]. It should be noted that segmentation of multidimensional time series is a separate task, and its solution goes beyond the consideration of this work.

Identifying patterns does not provide an unambiguous procedure for any types of data. Different clustering methods provide different group separation; moreover, the same clustering method can lead to different results with different initial initialization. Therefore, it is extremely important from the many existing clustering methods and similarity measures to choose those that best group the available data [34]. This is one of the reasons for the use of many approaches to identifying patterns and their grouping in honeynets.

3.2. The autoencoder architecture

Pre-processing of time series is performed on the basis of the neural network recurrent autoencoder (Fig. 1).

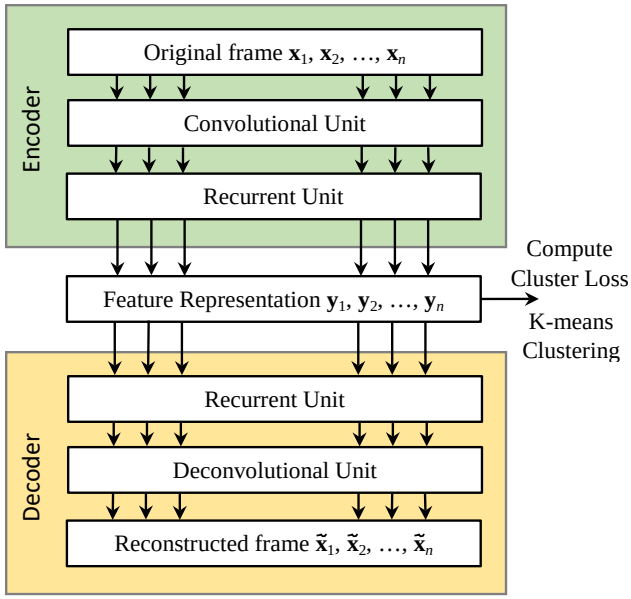

Figure 2: Recurrent Autoencoder Architecture

The autoencoder consists of a recurrent encoder and a recurrent decoder. The recurrent encoder consists of a convolutional unit (Fig. 2, a) and a recurrent unit (Fig. 3). The convolutional unit includes a convolutional layer with a convolution kernel n×1 (n = 3, 5, 7 – a size of a kernel along time axis, thus convolution performs along a time axis), maxpool layer with a kernel 2×1 (2 - size of a kernel along time axis) and batchnormalization layer. Maxpool laeyr is used in the case of reducing

the number of samples before processing by the recurring block. Using a 2×1 kernel halves the number of points along the time axis.

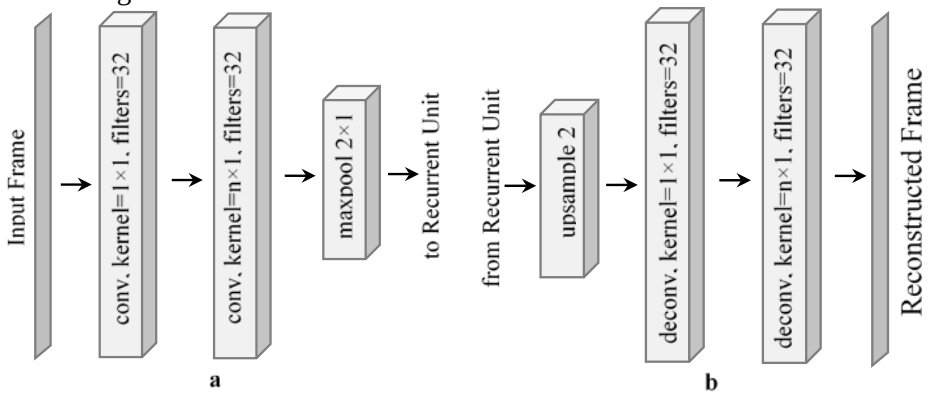

Figure 3: Convolutional (a) and Deconvolutional (b) Units

The Recurrent Unit is a stack of *m* LSTM layers with residual connections (Fig. 3). LSTM is a modification of the RNN architecture, which is useful for modeling given sequences. RNN training to account for long-term dependencies are a very difficult task due to the problem of a vanishing and exploding gradient. One solution to eliminate the problem of the exploding gradient is to use the clipping gradient, that is, reducing the gradient that exceeds the determined threshold value [35]. Adding residual connections adds flexibility to the neural network to avoid vanishing or exploding gradient [36]. Since the gradient is transmitted through the layers, this allows increasing the number of layers in the recurrent block and improves the representation of features at its output. At the output of the last layer of this stack, a representation of the features of the initial time series is obtained. The information signal of the stack of LSTM layers from each input $x(t)$ to each hidden layer and from each hidden layer to the output $y(t)$ passes in accordance with the ratios:

$$x'(t,k) = x(t) \| y'(t,k-1), y(t) = y'(t,1) \| \ldots \| y'(t,K),$$
$$i(t,k) = (W_{kxi} x'(t,k) + W_{khi} h(t-1,k) + W_{kci} c(t-1,k) + b_{ki}),$$
$$f(t,k) = \left(W_{kxf} x(t) + W_{khf} h(t-1,k) + W_{kcf} c(t-1,k) + b_{kf}\right),$$
$$c(t,k) = f(t,k) c(t-1,k) + i(t,k) \tanh(W_{kxc} x'(t,k) + W_{khc} h(t-1,k) + b_{kc}),$$
$$o(t,k) = (W_{kxo} x'(t-1,k) + W_{kco} c(t,k) + b_{ko}),$$
$$h(t,k) = o(t,k) \tanh^{-1}\left(c(t,k)\right),$$
$$y(t,k) = W_{kyh}(t,k) + b_{ky}.$$

here || denotes concatenation of vectors; $h(t,k)$ is a hidden state of a layer k for time moment t; i, f, o are the input, forget, and output gates respectively. This representation displays the characteristics of the initial time series and is used for further clustering, as well as for calculating the component of the clustering loss function (1).

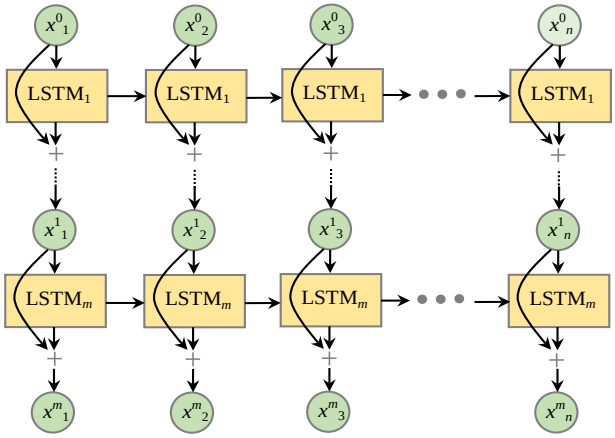

Figure 4: Recurrent Unit

3.3. The loss function of the autoencoder

The loss function for encoder training consists of two components: the loss function for reconstruction of the input signal by the decoder and the loss function for clustering

$$L(\theta) = L_r(\theta) + L_c(\theta). \quad (1)$$

The reconstruction loss function by the decoder $L_r(\theta)$ belongs to the loss functions used for regression problems. Functions that are offered to use for training of the recurrent autoencoder:

-mean squared error

$$L_{MSE}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2, \quad (2)$$

here y_i is a target value; $\tilde{y}_i$ is a value of neural network model output.

-mean absolute error

$$L_{MAE}(\theta) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i|. \quad (3)$$

-Huber loss function

$$L_H(\theta) = \frac{1}{n}\sum_{i=1}^{n} l_\delta,\ l_\delta = \begin{cases} \frac{1}{2}(y_i - \tilde{y}_i)^2, & |y_i - \tilde{y}_i| \le \delta, \\ \delta|y_i - \tilde{y}_i| - \frac{1}{2}\delta^2, & otherwise, \end{cases} \quad (4)$$

here δ is a threshold value determining which error to consider sufficiently small.

-log-cosh loss

$$L_{LHC}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\log\big(\cosh\big((y_i - \tilde{y}_i)\big)\big). \quad (5)$$

As well as the reconstruction loss function, the clustering loss function $L_c(\theta)$ can be selected from a specific list depending on the clustering method and the nature of the similarity measure between objects. K-means clustering is chosen as the clustering method due to the comparable simplicity of its implementation and the comparable speed of calculations that this method can provide. The previous preprocessing of the original time series by a recurrent autoencoder and the use of an appropriate measure of similarity ensures that the relationships between the time series of data collected by the honeynet are correctly identified. The main disadvantage of the k-means clustering method is the need to have preliminary information about the number of clusters. However, there is no exact information on the number of types of attacks that occur or will occur in the network. Therefore, the number of clusters should be determined in the clustering process.

Loss function for k-means clustering

$$L_c(\theta) = \sum_{j=1}^{N}\sum_{k=1}^{K} s_{jk} \cdot d\big(x_j, \mu_k\big), \quad (6)$$

here x_j is an object; μ_k is a centroid of a cluster k; s_{jk} is a boolean value that represent the membership of the object x_j to a cluster with a centroid μ_k; $d\big(x_j, \mu_k\big)$ is a measure of similarity between an object x_j and a centroid μ_k.

The measure of similarity used for clustering time series includes [37]:

-Euclidean distance

$$d_e(x, \mu) = \sqrt{\sum_{i=1}^{n}(x_i - \mu_i)^2}, \quad (7)$$

here n is the number of observations. It should be noted that the Euclidean distance is invariant with respect to features in the time domain, so this distance is difficult to apply to the original time series.

-Pearson correlation coefficient

$$d_r(x, \mu) = \frac{n\sum_{i=1}^{n} x_i\mu_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} \mu_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} \mu_i^2 - \left(\sum_{i=1}^{n} \mu_i\right)^2}}. \quad (8)$$

-Mahalanobis distance

$$d_m(x, \mu) = \sqrt{(x - \mu)^T \cdot C^{-1}(x - \mu)}, \quad (9)$$

-Distance obtained by dynamic time warping algorithm [38]. This distance is a mapping of two vectors and can be applied to vectors of different lengths $\mathbf{x} = x_1, x_2, \dots, x_n$ and $\boldsymbol{\mu} = \mu_1, \mu_2, \dots, \mu_n$. The matrix of dimension $n \times m$ contains the distances $\big(x_i, \mu_j\big)$ between two points.

The warped path $W = w_1, w_2, \dots, w_m$ $((m, n) \le k < m + n + 1)$ is formed in accordance with rules: a) limit condition: $w_1 = C(1,1),\ w_K = C(n, m)$; б) monotonicity condition:, , $a \ge a'$, $b \ge b'$;

в) step size condition: $w_k = C(a,b)$, $w_{k-1} = C(a',b')$, $a - a' \leq 1$, $b - b' \leq 1$. Among the set of paths corresponding to the above conditions, the one corresponding to the minimum ratio is selected:

$$d_w(x,\mu) = \min \sqrt{\sum_{k=1}^{K} w_k}, \quad (10)$$

3.4. Training and feature extraction with recurrent autoencoder

Clustering time series which represent honeypots data using a recurrent autoencoder includes the following steps:

1) Collection and preprocessing of data including actually data collection by the honeynet; selection of time window duration; scaling.

2) Training of a recurrent autoencoder, which consists of two stages: initial training and tuning. The initial training of the autoencoder involves training the model without taking into account the final loss of clustering and aims to form initial representations of the features of the time series. The loss function (1) in the 1st learning step has only one component – the reconstruction decoder loss function, which is selected from (2)-(5):

$$L(\theta) = L_r(\theta).$$

The autoencoder tuning is performed taking into account the clustering loss function (6), one of the similarity measures (7) - (10) is used. The autoencoder loss function takes the form (1). To calculate the clustering loss function, the features representation $\mathbf{y} = y_1, y_2, \dots, y_m$ obtained at the output of the encoder is used. Model tuning is a process of continuous (periodic) training of the model, takes into account current changes in the dataset.

3) Feature extraction. A recurrent autoencoder encoder is used to obtain a feature representation.

4) A clustering procedure, which provides next operations:

a) Calculate the clustering algorithm for different cluster quantity N_C values from 1 to N_{Cmax}; here N_C - the number of clusters, N_{Cmax} - the maximum number of clusters.

b) Calculate the integral distance criterion for each N_C between objects within a cluster $J = \sum_{i=1}^{l} d(C_l)/l$; here $d(C_l)$ is the distance between objects within one cluster; l is the number of clusters. This distance for the cluster $C_i = \{X_1, X_2, \dots X_m\}$ is determined by the relationship $d(C_i) = \sum_{a=1}^{m} \sum_{b=1}^{m} d(X_a, X_b)/m$; here $d(X_a, X_b)$ is distance between two objects, which is determined by one of the relations (6)-(9) depending on which measure of similarity was chosen for clustering.

c) Selection from the set of calculated integral criteria $\{J_{NC1}, J_{NC2}, \dots\}$ the minimum one and corresponding to it the number of N_C and the set of centroids $\boldsymbol{\mu}^1(\mu_1^1, \mu_2^1, \dots \mu_n^1)$, …, $\boldsymbol{\mu}^{N_C}(\mu_1^{N_C}, \mu_2^{N_C}, \dots \mu_n^{N_C})$. Cluster centroids represent patterns of attacks that operate in the honeynet.

The clustering process is periodically repeated to continuously update attack patterns. Accordingly, steps 1 to 4 of the clustering procedure are repeated. It is possible to exclude the initial training of the autoencoder from the update procedure, and in the case of critical computation time requirements, this exception is advisable.

4. Results & discussion

The analysis of data of the honeynet on the basis [39-41] was conducted. For analysis, a dataset was created which contained time series of data: the number of malicious sources per time unit; the number of new malicious sources per time unit; the volume of data received per time unit; the volume of data sent per time unit; the volume of data per message; the number of sessions per time unit; session duration; “lifetime” of a malicious source (Fig. 5). A fixed size window and a window derived from the previous segmentation were used to determine the length of the time series. Clustering was performed using different combinations of an autoencoder reconstruction loss function and a clustering similarity measure. The results were comparable to the pre-labeled patterns and are shown in Table 1. The best results among similarity measures are shown by dynamic time warping distance, which is quite logical, since this distance is more invariant to some displacements compared to other similarity measures. However, when using this similarity measure, the calculation time is the longest, which should be correlated with the performance requirements.

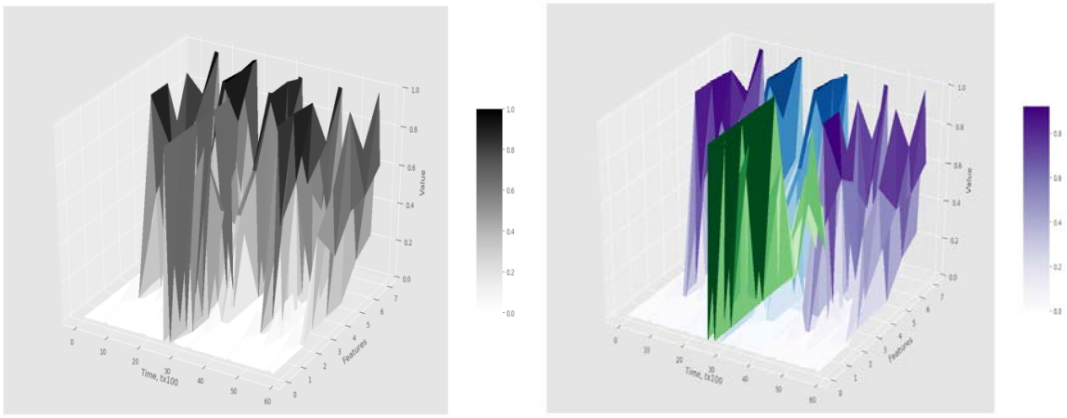

Figure 5: Initial and Clustered Fragment of Time Series Represented Honeynet Data

Table 1

Clustering Accuracy Values (percentage)

Autoencoder Loss Function \ Clustering Similarity Measure	Euclidian Distance	Pearson correlation coefficient	Mahalanobis distance	Dynamic Time Warping Distance
Mean Squared Error	69	79	71	83
Mean Absolute Error	69	80	70	83
Huber Loss Function	70	82	71	84
Log-cosh Loss	67	81	69	81

5. Conclusions

The use of honeypots and honeynets to collect and analyze the characteristics of the activity of attackers allows to detect new attacks in addition to the known ones, to find among them attacks with similar characteristics, and accordingly to identify patterns of new attacks. A properly designed honeynet allows tracking malicious actions at all points in a working computer network. The information provided by such a honeynet, together with the corresponding analysis, allows real-time monitoring of malicious actions, clustering them, obtaining patterns of attacks to which the network is exposed, and therefore increases the security of the computer network in general.

In the paper, it is proposed to use time series of attackers' activity to identify patterns of attacks, which are formed on the basis of data collected by a honeynet. To obtain a representation of the characteristics of multidimensional time series, it is proposed to use the recurrent autoencoder, which is built on the basis of convolutional and recurrent units. The advantage of using a recurrent block is that it effectively processes the features of multidimensional time series. The recurrent unit is a stack of LSTM layers with residual connections, which helps to avoid the vanishing and exploding gradient when optimizing the weights. The clustering loss function together with the autoencoder loss function is used to optimize the recurrent autoencoder, which generally increases the accuracy of clustering. Because of such preprocessing of time series with a recurrent autoencoder, it is possible to use not only measures of similarity ordinary for time series (dynamic time warping, Pearson correlation coefficient), but also Euclidean distance and Mahalanobis distance.

Further research on modifications of the autoencoder architecture is planned to improve the representation of time series features. It is also planned to investigate the methods of segmentation of multidimensional time series and their impact on the accuracy of clustering.

6. References

[1] S. Tomas, Study of Internet Threats and Attach Methods Using Honeypots and Honeynets, Computer Networks 431 (2014) 118–127.

[2] T. Sochor, M. Zuzcak, Attractiveness Study of Honeypots and Honeynets in Internet Threat Detection, Computer Networks 522 (2015) 69-81. doi: 10.1007/978-3-319-19419-6 7.
[3] S. Lysenko, K. Bobrovnikova, S. Matiukh, I. Hurman, O. Savenko, Detection of the botnets' low-rate DDoS attacks based on self-similarity, International Journal of Electrical and Computer Engineering 10 4 (2020) 3651-3659
[4] O. Savenko, A. Nicheporuk, I. Hurman, S. Lysenko, Dynamic signature-based malware detection technique based on API call tracing, CEUR-WS 2393 (2019) 633-643
[5] S. Lysenko, K. Bobrovnikova, O. Savenko. A Botnet Detection Approach Based on The Clonal Selection Algorithm, in: Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DeSSerT-2018, Kyiv, Ukraine, 2018, pp. 424-428.
[6] S. Lysenko, K. Bobrovnikova, O. Savenko and A. Kryshchuk, BotGRABBER: SVM-Based Self-Adaptive System for the Network Resilience Against the Botnets' Cyberattacks, Communications in Computer and Information Science, 1039 (2019) 127-143. doi: 10.1007/978-3-030-21952-9_10
[7] L. Portnoy, E. Eskin, S. Stolfo. Intrusion Detection with Unlabeled Data Using Clustering. Proceedings of ACM. CSS Workshop on Data Mining Applied to Security (DMSA2001. Citeseer, 2001.
[8] J. Mazel, P. Casas, P. Owezarski. Sub-Space Clustering and Evidence Accumulation for Unsupervised Network Anomaly Detection. Proceedings of the Third International Conference on Traffic Monitoring and Analysis, ser. TMA'11. Berlin: Springer-Verlag, 2011, pp. 15–28.
[9] N.A. Rosli, W. Yassin, M.F. Faizal, S.R. Selamat. Clustering Analysis for Malware Behavior Detection using Registry Data. (IJACSA) International Journal of Advanced Computer Science and Applications, 10 12 (2019) 93-102
[10] M. Ahmed, A. N. Mahmood. Novel Approach for Network Traffic Pattern Analysis using Clustering-based Collective Anomaly Detection, Annals of Data Science, 2 (2015)
[11] S. Taheri, A.M. Bagirov, I. Gondal, S. Brown. Cyberattack triage using incremental clustering for intrusion detection system. Internation Journal of Information Security, 19 (2020) 597–607. doi: https://doi.org/10.1007/s10207-019-00478-3.
[12] M. Landauer, F. Skopik, M. Wurzenberger, A. Rauber. System log clustering approaches for cyber security application: A survey. Computers & Security 92 (2020) 101739 1-17.
[13] S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk and K. Bobrovnikova DNS-based Anti-evasion Technique for Botnets Detection, in: Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw, 2015, pp. 453–458.
[14] O. Savenko, S. Lysenko, A. Kryschuk, Multi-agent based approach of botnet detection in computer systems Communications in Computer and Information Science, 291, 2012, 171-180
[15] S. Lysenko, O. Savenko, K. Bobrovnikova, A. Kryshchuk Self-adaptive System for the Corporate Area Network Resilience in the Presence of Botnet Cyberattacks, Communications in Computer and Information Science, 860 (2018). doi: https://doi.org/10.1007/978-3-319-92459-5_31.
[16] R. Kozma, J. L. G. Rosa and D. R. M. Piazentin Cognitive clustering algorithm for efficient cybersecurity applications. The 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, 2013, pp. 1-8, doi: 10.1109/IJCNN.2013.6706774.
[17] E. Min, X. Guo, Q. Liu, G. Zhang, J. Gui, J. Long. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture IEEE Access, vol. 6, pp. 39501-39514, 2018, doi: 10.1109/ACCESS.2018.2855437
[18] M. Kampffmeyer, S. Løkse, F. M. Bianchi, L. Livi, A. Salberg, R. Jenssen, Deep Divergence-Based Approach to Clustering. Neural networks : the official journal of the International Neural Network Society, 113 (2019) 91-101.
[19] J. Zhuxi, Z. Yin, T. Huachun, T. Bangsheng, Z. Hanning, Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering, arxiv.org 1611.05148.
[20] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. https://arxiv.org/abs/1606.03657.

[21] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 1096–1103.
[22] K. Zeng, J. Yu, R. Wang, C. Li and D. Tao, "Coupled Deep Autoencoder for Single Image Super-Resolution," in IEEE Transactions on Cybernetics, vol. 47, no. 1, pp. 27-37, Jan. 2017, doi: 10.1109/TCYB.2015.2501373.
[23] K. Sohn, H. Lee, X. Yan, Learning structured output representation using deep conditional generative models, in: Advances in Neural Information Processing Systems, 2015, pp. 3483–3491.
[24] K. Sohn, H. Lee, X. Yan, Learning structured output representation using deep conditional generative models, in: Advances in Neural Information Processing Systems, 2015, pp. 3483–3491.
[25] Y. Raut, T. Tiwari, P. Pande, P. Thakar, Image Compression Using Convolutional Autoencoder. In: Kumar A., Paprzycki M., Gunjan V. ICDSMLA 2019. Lecture Notes in Electrical Engineering, vol 601. Springer, Singapore. https://doi.org/10.1007/978-981-15-1420-3_23.
[26] M. Aamir, N. M. Nawi, H. B. Mahdin, R. Naseem, M. Zulqarnain. Auto-Encoder Variants for Solving Handwritten Digits Classification Problem. International Journal of Fuzzy Logic and Intelligent Systems, 20 1 (2020) 8-16. doi: http://doi.org/10.5391/IJFIS.2020.20.1.8.
[27] Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, Stefan Decker, Deep learning-based clustering approaches for bioinformatics, Briefings in Bioinformatics, Volume 22, Issue 1, January 2021, Pages 393–415.
[28] M. Caron, P. Bojanowski, A. Joulin, M. Douze, Deep Clustering for Unsupervised Learning of Visual Features In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, 11218. doi: https://doi.org/10.1007/978-3-030-01264-9_9.
[29] D. Mautz, C. Plant, C. Böhm, DeepECT: The Deep Embedded Cluster Tree. Data Sci. Eng. 5, 419–432 (2020). doi: https://doi.org/10.1007/s41019-020-00134-0.
[30] S, A, Shah, V. Koltun, Deep Continuous Clustering, arxiv.org 1803.01449.
[31] P. Owezarski, A near real-time algorithm for autonomous identification and characterization of honeypot attacks, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ser. ASIA CCS '15. New York, NY, USA: ACM, 2015, pp. 531–542.
[32] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, J. Schönfelder. A Survey on Honeypot Software and Data Analysis, arxiv.org 1608.06249.
[33] X. Wang, J. W. Emerson. Bayesian Change Point Analysis of Linear Models on Graphs. arxiv.org 1509.00817.
[34] Jain A.K. Data clustering: 50 years beyond k-means Pattern recognition letters, vol. 31, no. 8, pp. 651–666, 2010.
[35] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013, 1310–1318.
[36] J. Kim, M. El-Khamy, J. Lee. Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition, arxiv.org 1701.03360.
[37] F. Iglesias, W. Kastner, Analysis of Similarity Measures in Times Series Clustering for the Discovery of Building Energy Patterns Energies 6 (2013) 579-597.
[38] Keogh E. Exact Indexing of Dynamic Time Warping, in: Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong, China, 20–23 August 2002; pp. 406–417.
[39] Traffic Data from Kyoto University's Honeypots, URL: https://www.takakura.com/Kyoto_data/.
[40] Lysenko, S., Savenko, O., Bobrovnikova, K., Kryshchuk, A., Savenko, B.: Information Technology for Botnets Detection Based on Their Behaviour in the Corporate Area Network. In: International Conference on Computer Networks, 2017, pp. 166-181. Springer, Cham.
[41] Drozd O., Kharchenko V., Rucinski A., Kochanski T., Garbos R., Maevsky D. Development of Models in Resilient Computing, Proc. of 10th IEEE International Conference on Dependable Systems, Services and Technologies (DESSERT' 2019), Leeds, UK, June 5-7 2019, pp. 2-7 (2019), doi: 10.1109/DESSERT.2019.8770035.