computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

BOTNET DETECTION APPROACH BASED ON THE DISTRIBUTED SYSTEMS

Oleg Savenko [1], Anatoliy Sachenko [2], Sergii Lysenko [1], George Markowsky [3], Nadiia Vasylkiv [2]

[1] Khmelnitsky National University, 29016, 11 Instytutska st., Khmelnytskyi, Ukraine, savenko_oleg_st@ukr.net, sirogyk@ukr.net
[2] Ternopil National Economic University, 46020, 11 Lvivska st., Ternopil, Ukraine, {as, nvs}@tneu.edu.ua
[3] Missouri University of Science and Technology, 65409, 300 W 13th St., Rolla, USA, markov@mst.edu

Paper history:
Received 18 March 2019
Received in revised form 20 May 2020
Accepted 21 May 2020
Available online 14 June 2020

Keywords:
malware;
botnet;
botnet detection;
distributed systems;
attacks;
naive Bayes classifier;
network security.

Abstract: The paper presents a botnet detection approach for the distributed systems. It is based on the developed three level model, which includes botnet's components: command and control center, control centers, basic elements of the botnet (bots). The novel framework provides the ability to detect known and unknown botnets, and consists of the host and the network levels. At the host level, the detection procedure is based on the implementation of the Bayes classification. The network level extends the results obtained at the host level to the rest of the local area network. Proposed approach provides the exchange of the results obtained by the Bayes classification for further use by other program units of the distributed system. The results of the developed classifier show that representation of the botnets' samples for different classes and subclasses is sufficient for efficient botnet detection. Proposed technique demonstrates promising results concerning botnet detection in the distributed systems.

1. INTRODUCTION

The trends concerning malware development and its spreading demonstrate an active extending of the malware's technical capabilities. The main motivating factors leading to its creation are the financial gain and political advantage. One of the most rapidly evolving directions of the malware is the botnets that allow an attacker to gain remote access of the user's computers. The harm caused by the use of such malicious software is rapidly increasing [1].

Architecture of the modern antiviruses has a single central control center. Such tools as ESET Endpoint Security for Windows Endpoint Security for corporate networks [2], Dr.Web CureNet! [3], Symantec Endpoint Protection [4], Malwarebytes Endpoint Security [5], Cisco® Network Admission Control (NAC) [6] are based on the centralized way of functioning. Kaspersky Administration Kit Antivirus is based on the principle of autonomous work, and the decision-making is implemented without the administrator participation in case of the critical situations. However, it is also based on a centralized way of organizing the interaction of system components [7]. Mentioned tools are based on methods that do not sufficiently take into account all stages of the botnets functioning and their possible structures, therefore it leads to the decreasing in the botnet detection efficiency.

Therefore, the development of new methods and tools for efficient botnets detection in the distributed systems is an urgent problem.

2. RELATED WORKS

In the recent years, the great number of the botnet detection approaches based on machine learning are developed [8, 9]. In [10] the botnet detection is based on the analysis of its group activity in the network. The behavior is analyzed by the histogram in order to determine the number of web requests and their diversity over time using HTTP bots. Proposed method uses the correlation analysis to detect botnets based on the similarity and the correlation of their group activities in the network.

The modeling system of the botnets' architectures via agents, which take into account various botnets' functioning mechanisms, is presented in [11]. It is based on the necessity to take into consideration the special aspects of the botnets' structure, as it is important for gathering the characteristics of the botnets. In the articles [12-16] the methods for botnets detection based on the traffic analysis and anomaly detection are presented. The disadvantage of the technique is the need for a constant traffic analysis and the obtaining needed features which can be changed rapidly by attackers. Moreover, technique does not take into account the botnets' architectures. In [17, 18] the botnet detection methods are based on signatures. Technique requires capturing of a great number of packages and their comparisons with the pre-configured attack templates from database. The common disadvantage of these methods is the need to update templates that affects the inability to detect new botnets or their nodes. In [19] a mechanism for analysis of botnet activities in the IoT, based on machine learning techniques is presented. It is based on network flow identifiers that can track suspicious activities of botnets. In [20-22] the methods for botnets detection in corporate area networks (CAN), which include the use of a multi-agent system, are presented. A botnet detection is based on the analysis of the botnets' behavior in the CAN. Method is able to detect bots, that use such evasion techniques as cycling of IP mapping, "domain flax", "fast flux" and DNS-tunneling. In [23] a system of baits for malware provoking, which is located in a distributed system, is developed. To identify new botnets, the system requires a permanent addition of malware's behaviors. In [24] a botnet detection approach uses the unsupervised machine learning and similarity analysis between benign traffic data and botnet's traffic data. Known methods and tools do not provide high efficiency of the botnet detection. This is due to the development of new techniques for the botnet distribution in the networks and computer systems and appearance of new capabilities of the botnet functioning. Moreover, network antivirus tools are mainly based on the rigidly centralized architecture, which is also used by intruders to attack the computers systems, which contain such center.

Therefore, the development of new effective methods and tools for botnet detection, which will take into account the perspective possibilities of botnets' functioning and the distributed architecture is an actual problem.

3. PROPOSED TECHNIQUE FOR BOTNET DETECTION

3.1 THE STRUCTURE OF THE CONTROLLED DISTRIBUTED BOTNET

The botnet is a distributed software system that includes a great number of nodes (bots), which communicate via malicious software. Structurally botnets include the nodes, which are assigned to control of the network and maintain its integrity, and the end nodes are aimed at carrying out the malicious actions. An attacker through a command-control center (C&C) or via other intermediate remote control centers [20, 25, 26] directly controls the botnet.

Let us define the botnet's components: command and control center, control centers, basic elements of the botnet (bots). The structure of the botnet is shown in Fig. 1. Let us define the basic elements of the botnet as the subset E_{3,i_3} $i_3 = 1, 2, \dots, n_3$, the botnets' control centers as the subsets E_{2,i_2}, where $i_2 = 1, 2, \dots, n_2$ – a number of botnet's basic elements, where n_2 – is a number of control centers of the botnet.

Let us define the command-control centers of the botnet as the subsets E_{1,i_1} where $i_i = 1, 2, \dots, n$, n_1 – a number of command-control centers of the botnet. The number of these centers may vary.

Botnets may have different architectures, depending on the topology and communication elements: multi-server, hierarchical, random (peer-to-peer) and hybrid. Presented architecture in Fig. 1 is generalized and covers these topologies.

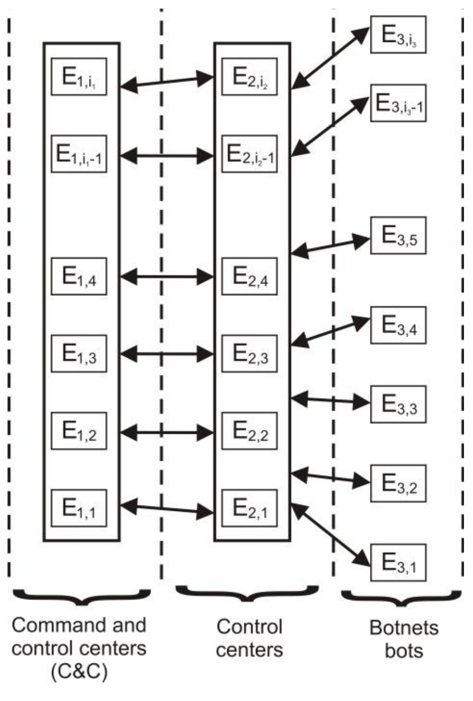

Figure 1 –The structure of the controlled distributed botnet

Let us present the botnet as the union of its components as follows:

$$E = \bigcup_{i_1=1}^{n_1} E_{1,i_1} \bigcup_{i_2=1}^{n_2} E_{2,i_2} \bigcup_{i_3=1}^{n_3} E_{3,i_3}, \qquad (1)$$

where E – a set of the botnet's components. As the elements of the subsets $E_{w,i}$ are the functions f_{i_1,i_2,i_3}. Different elements of subsets $E_{w,i}$ may include the same functions. Functional load of each of the assigned functions depends on the type of operating systems and their API functions, respectively.

Let us consider the botnet's malicious action as the sequence of the API calls.

In order to represent botnet's behavior malicious actions let us define its main items:

- a vector that describes its malicious actions $v_{z,f_{i_1,i_2,i_3}}$ and corresponding vectors $v_{API,x,f_{i_1,i_2,i_3},m_u,l}$ that describe its malicious actions, presented via the variants of API functions that are able to execute malicious actions;
- vectors of possibly malicious actions $v_{p,e_s} = \left(v_{p,e_s,1}, v_{p,e_s,2}, \dots, v_{p,e_s,n_{v_p}}\right)$ captured at the monitoring stage and corresponding vectors $v_{p,K,e_s} = \left(v_{p,K,e_s,1}, v_{p,K,e_s,2}, \dots, v_{p,K,e_s,n_{v_p}}\right)$ that describe its malicious actions, presented via the variants of API functions captured at the monitoring stage
- z defines the malicious action;
- x – a type of operating system;
- p – possible malicious action;
- m_u – a number of vector's component;
- l – a number of variations of malicious action presentation via API functions;
- e_s - the vector number;
- s – a number of variant of the malicious action presentation via API functions;
- n_{v_p}- a number of the vector components v_{p,e_s}.

The vector components v_{p,e_s} are the numbers of API functions that may be executed by the botnets. The task of the classifier is to assign the analyzed vector v_{p,e_s} to one of the botnets' classes.

Based on the structure presented in Fig. 1 of the reference botnet's model, presented via the vectors of the malicious actions is assigned to class K_i. Known types of the botnets are characterized by the different functional possibilities. Furthermore, some malicious actions, implemented via API functions, may occur more frequently.

One malicious botnet's action may be described by more than one vector, which contain the sets of the most often called API functions to perform specified malicious botnet's actions. The botnets' classes are characterized by mentioned vectors. Because the structure of the botnet may include the basic elements, control centers and command-control centers of the botnet and corresponded vector of malicious actions could not be compared with the whole class, each class could be divided into subclasses.

In order to establish, that the resulting vector is a malware or benign software, the naive Bayesian classifier is used. Its main benefits are: simple and easy implementation, it does not require as much training data, it handles both continuous and discrete data, it is highly scalable with the number of predictors and data points, it is fast and can be used to make real-time predictions, and it is not sensitive to irrelevant botnets' features.

Let us define $V_p = \left\{v_{p,1}, v_{p,2}, \dots, v_{p,n_{V_p}}\right\}$ as the sample formed on the basis of the API calls for the vectors v_{p,e_s}, A – a hypothesis about the membership of values V_p to one of the botnets classes K_l, where $l = 0, 1, \dots, 6$. In order to solve the classification problem we are to evaluate the probability, that the sample V_p belongs to the class K_l, taking into account the knowledge about the botnets' actions. For this purpose we need to define the probability $P(\mathrm{A} \mid V_p)$ that the hypothesis A contains the data from the sample V_p. Let us evaluate a posteriori probability $P(\mathrm{A} \mid V_p)$ - the probability that the value of A depends on the actions of the sample V_p, using Bayes' theorem.

Each botnets class K_l is defined and represented by a set of pairs vectors $v_{z,f_{i_1,i_2,i_3}}$ and $v_{API,x,f_{i_1,i_2,i_3},m_u,l}$. The sample V_p belongs to the class K_l with the highest value of the posteriori probability if and only if the condition is fulfilled:

$$P(K_{l_1} \mid V_p) > P(K_{l_2} \mid V_p), \qquad (2)$$

For all l_1 and l_2, such that $0 \le l_1 \le 6,\ 0 \le l_2 \le 6, l_1 \ne l_2$.

So, we search for a class with the highest value of the probability $P(K_l \mid V_p)$.

In order to assign the vector v_{p,e_s} to the botnet's certain class, the product of the probabilities of API functions that were included into the vector of potentially suspicious actions is to be evaluated. For this purpose, the multi-nominal generative model, which takes into account the number of repetitions of API functions and does not take into account the absence of some API functions, was used.

The definition of the membership of the vector v_{p,e_s} to the class K_y or its subclass $K_{y,g}$ is performed on the basis of the calculations of the probabilities for each class or subclass using Bayes classifier evaluations:

$$\mathrm{P}\left(\mathrm{v}_{\mathrm{p},\mathrm{e}_\mathrm{s}} \mid \mathrm{K}_{\mathrm{y},\mathrm{g}}\right) =$$
$$= \ \mathrm{P}\left(\left|\mathrm{n}_{\mathrm{v}_\mathrm{p}}\right|\right) \mathrm{n}_{\mathrm{v}_\mathrm{p}}! \ \times$$
$$\times \ \prod_{\mathrm{w}=1}^{\mathrm{n}_{\mathrm{m}_\mathrm{u}}} \left(\frac{1}{\mathrm{v}_{\mathrm{p},\mathrm{K},\mathrm{e}_\mathrm{s},\mathrm{w}}!} \left(\frac{\mathrm{v}_{\mathrm{API},\mathrm{x},\mathrm{w},\mathrm{f}_{\mathrm{i}_1,\mathrm{i}_2,\mathrm{i}_3},\mathrm{g},\mathrm{l}}}{\sum_{\mathrm{g}=1}^{\mathrm{m}_{\mathrm{u},0}} \mathrm{v}_{\mathrm{API},\mathrm{x},\mathrm{w},\mathrm{f}_{\mathrm{i}_1,\mathrm{i}_2,\mathrm{i}_3},\mathrm{g},\mathrm{l}}} \right)^{\mathrm{v}_{\mathrm{p},\mathrm{K},\mathrm{e}_\mathrm{s},\mathrm{w}}} \right) \tag{3}$$

In order to conduct the training procedure, the probabilities $P\left(v_{z,z_v,f_{i_1,i_2,i_3}} \mid K_{y,g}\right)$ are to be processed. For this purpose, we evaluate the optimal estimates of the probabilities that some API function will be present in each class or subclass by modifying the result using the Laplace algorithm to avoid the "zero-frequency" problem:

$$P\left(v_{z,d,f_{i_1,i_2,i_3}} \mid K_{y,g}\right) =$$
$$= \frac{1 + \sum_{b=1}^{\sum_{g=1}^{m_{u,y}} l_{g,y}} v_{API,x,b,f_{i_1,i_2,i_3},g,l}\, P\left(K_{y,g} \mid v_{z,b,f_{i_1,i_2,i_3}}\right)}{n_{m_u} + \sum_{d=1}^{n_{m_u}} \sum_{b=1}^{\sum_{g=1}^{m_{u,y}} l_{g,y}} v_{API,x,d,f_{i_1,i_2,i_3},g,l}\, P\left(K_{y,g} \mid v_{z,b,f_{i_1,i_2,i_3}}\right)} \tag{4}$$

3.2 LEARNING PROCEDURE

The learning procedure involves the following stages:

1) definitions of the subclasses via one presentation of API functions for each of them, and calculation of the probabilities for each of the subclasses and classes, and their definitions as primary;

2) for each known next variation of the presentation of the malicious botnet's action via vector of API function, is to be classified by the Bayes classifier into the classes and subclasses;

if the received presentation of the malicious botnet's action is assigned correctly to the specified subclass, then its marked elements are added to the subclass as a separate sample;

if obtained result does not classify it into the required subclass, it in this subclass, but at the same time comparisons with other values of the classes are to be performed; the result of the comparisons will be the deviation from initial values of probabilities;

if the resulting probability is significantly (the threshold more than 10%) different from the primary probability of a subclass or class, then a new separate subclass of this class is to be created;

for each learning stage the probabilities for each API function of the subclasses and classes are to be calculated; for those subclasses and classes, where the divergence with the primary classes is more than 10%, a new subclass is to be created in its subclass;

all probabilities obtained after several iterations are averaged and are considered as appropriate probabilities for use in further calculations;

3) after the basic training phase is completed and the new data to Bayes classifier is added, the deviations verification for the additional subclasses is to be performed, and divergence between its mean probabilities values is to be evaluated;

if the divergence value is less than 10%, then the subclass is added by the data of the additional subclass, and all probabilities and their mean values are to be recalculated;

4) at each stage the difference between the mean probabilities' values and the difference between the probabilities values obtained by the classifier are to be evaluated; if the difference is more than 10% for some subclasses, then training is to be continued for them by adding additional data and repeating the steps 1-3.

The vector v_{p,e_s} may not be assigned to any given class and subclass. It means that the analyzed object does not include probably malicious actions. This fact is based not only on the search for the maximum probability calculated by Bayes theorem, but also on the correspondence between this probability and the thresholds' values of classes and subclasses defined during the classifier's learning process. This is due to the fact that the executable process represented by the vector v_{p,e_s}, may belong to benign software. In this case, further analysis is interrupted.

The self-learning procedure is carried out according to the training scheme (steps 1-4). After the analysis of the vector of possible malicious actions is performed, its data has to be added to the class and its probabilities are to be calculated. If the deviations of the probabilities are within the specified thresholds for one of the subclasses, then after the classification is completed, its new data in its subclass is included and new calculation for the entire classifier and its mean values of probability deviations is performed.

After the new item is added to the classifier of the program unit, the obtained information is sent to other program units of the distributed system for use.

The decision concerning the location of the processing of the obtained vector of the malicious actions is determined on the basis of the computer system's workload, in which these data were collected. If the workload is high, then obtained vectors are sent to other program units for processing. After the data processing is complete, obtained results are to be sent back.

If the classifier analyzed software is assigned as the malicious, it is added to the classifier and is sent to all program units of the distributed system.

3.3 THE STAGES OF THE BOTNET DETECTION APPROACH BASED ON THE DISTRIBUTED ARCHITECTURE

The botnet detection approach based on the distributed architecture involves the stages:

1. Obtaining the information concerning the active processes using an active monitoring (starting from the first API function of each process that will be performed after the start of the computer system).

2. Gathering the monitoring data into the vector after detecting possible malicious actions in the computer system.

3. Formation of feature vector based on determined potentially malicious actions. The components the feature vector are the API functions.

4. The decision making about the location of the feature vector processing.

5. If the computing load of the computer system is low, then information is processed on this computer system, otherwise it is sent to another specified program unit of the computer system.

6. The implementation of the vector classification and analysis of its results.

6.1. If the feature vector has been assigned to one of the botnets' class, then this information is to be sent to all classifiers of all program units.

6.2. If the feature vector has been assigned to the several botnets classes, then other program units of the distributed system are to be involved for feature vector analysis.

6.3. If the similarity with the available botnets' classes is low, but other program units of the distributed system have made a decision that feature vector contains malicious action, a new botnet class is to be created, the classification information is to be updated and sent to all program units of the distributed system.

6.4. If the analyzed feature vector does not contain malicious actions, then the analysis is completed.

6.5. If the feature vector corresponds to malicious behavior, the analyzed executable is stopped.

6.6. Search for the similar processes in other computer systems of the network using installed program units of the distributed system based on the obtained information.

7. Calculation of the probability values for each program unit of the antivirus distributed system that the computer system is infected.

Thus, the developed technique is able to detect new botnets and is based on the distributed architecture and with the use of the Bayes classifier.

The architecture of the distributed system is presented in Fig. 2.

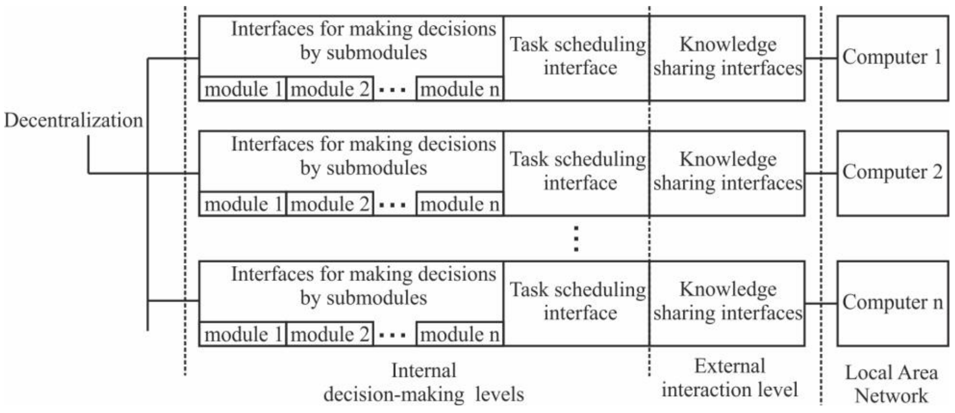
Decentralization
Interfaces for making decisions by submodules
module 1
module 2
module n
Task scheduling interface
Knowledge sharing interfaces
Computer 1
Computer 2
Computer n
Internal decision-making levels
External interaction level
Local Area Network

Figure 2 –The architecture of the distributed system

5. EXPERIMENTS

The purpose of the experiments was to verify the efficiency of the botnet detection technique using the Bayes classifier. In order to carry out the experiments, 28 artificial botnets were constructed, grouped by classes. Mentioned botnet had the functional properties of the bots' classes: Agobot, SDBot, Spybot, evilbot, DSNX, G-sys (remote control, usage of the system vulnerabilities, server attacks, system spying, etc).

Obtained malicious programs included 25 structural elements with three functioning stages which used 81 API functions [20]. Not all botnets used for experiments contained all possible structural elements and functions.

Each malicious sample was presented as the vector taking into account the variations of its presentation via API functions, and all samples are assigned into the botnets and classes and subclasses.

In order to conduct the experiment, the local network with 19 computer systems was employed. Each of the computer system contained the program unit without any other antivirus tools.

First, a program unit used a classifier with no one of the generated botnets' samples. One computer system contained the command control center, and the control centers were located in 3 computer systems, and 15 computer systems were infected via botnet's bots.

The installation of the generated botnets was carried out alternately. The experiment being completed, all computer systems in the network were completely updated, except the classifiers.

Each experiment for each botnet's sample lasted 96 hours. For the experiment, the botnets were selected that use the strategy of obtaining complete control over the computer system.

The experiments involved the extraction of the vectors of possible malicious actions via API calls monitoring in the computer system. Obtained vectors were analyzed by the classifier of the program unit. The experiments were carried out concerning the trained and untrained classifier.

The aim of the experiments was to determine the rates of the botnet detection efficiency for classes and subclasses using the Bayes classification.

In order to evaluate the method efficiency, let us consider its main parameters:

$P_{1,1}$ and $P_{1,2}$ - the rates of correctly classified vectors of botnets' malicious samples concerning botnets' classes using trained and untrained classifiers respectively;

$P_{2,1}$ and $P_{2,2}$ - the rates of correctly classified vectors of botnets' malicious samples concerning botnets' subclasses using trained and untrained classifiers respectively;

$P_{3,1}$ and $P_{3,2}$ – the rates of correctly identified computer systems as infected using trained and untrained classifiers respectively;

$P_{4,1}$ and $P_{4,2}$ – false positives (for trained and untrained classifiers);

$P_{5,1}$ and $P_{5,2}$– the rates of the malicious samples assigned to wrong botnet's class using trained and untrained classifiers respectively.

The results of the experiment for seven botnets classes are presented in Table 1.

Table 1. The results of the experiment

Parameter	**Resulting values for different classes**							**Average value**
	Class 0	***Class 1***	***Class 2***	***Class 3***	***Class 4***	***Class 5***	***Class 6***	
$P_{1,1}$, %	90,74	84,29	73,66	86,30	94,04	94,18	96,60	89,44
$P_{1,2}$, %	75,93	63,57	60,22	70,32	68,77	67,60	69,36	67,71
$\lvert P_{1,1}-P_{1,2}\rvert$, %	14,81	20,72	13,44	15,98	25,27	26,58	27,24	21,73
$P_{2,1}$, %	85,80	83,57	72,58	85,39	98,88	93,92	96,60	88,42
$P_{2,2}$, %	74,69	63,57	59,14	70,32	67,37	66,58	67,66	66,80
$\lvert P_{2,1}-P_{2,2}\rvert$, %	11,11	20	13,44	15,07	31,57	27,34	28,94	21,62
$P_{3,1}$, %	92,11	84,21	71,93	89,47	90,53	88,42	93,68	87,72
$P_{3,2}$, %	76,32	57,89	63,16	64,91	71,58	54,74	75,79	65,89
$\lvert P_{3,1}-P_{3,2}\rvert$, %	15,79	26,32	8,77	24,56	18,95	33,68	17,89	21,83
$P_{4,1}$, %	7,89	14,47	28,07	10,53	7,37	11,58	6,32	11,70
$P_{4,2}$, %	21,05	40,79	36,84	31,58	24,21	44,21	22,11	31,97
$\lvert P_{4,1}-P_{4,2}\rvert$, %	13,16	26,32	8,77	21,05	16,84	32,63	15,79	20,27
$P_{5,1}$, %	0	1,32	0	0	2,11	0	0	0,49
$P_{5,2}$, %	2,63	1,32	0	3,51	4,21	1,05	2,11	2,14
$\lvert P_{5,1}-P_{5,2}\rvert$, %	2,63	0	0	3,51	2,1	1,05	2,11	2,13

The results of the experiment demonstrated, that the accuracy of the botnet's samples classification is 66% for the classifier without involvement of botnet's samples and 88% for the classifier, which was trained using the generated vectors. The obtained results were averaged and their dispersion relative to the mean value is 1%.

The difference of deviation for each class and subclass using trained and untrained classification execution was 21.5%. The dispersion for each class and subclass using two ways of classification was less than 5%. The rates of false positives using two ways of classification execution were 11.7% (trained classifier) and 31.97% (untrained classifier).

The rates of assignment of the malicious samples to wrong botnet's class using trained and untrained classifiers were 0.01% and 2.14% respectively.

6. CONCLUSION

The paper presents a botnet detection approach for the distributed systems. The novel framework provides the ability to detect botnets. It consists of two parts: the host and the network levels.

At the host level, the detection procedure is based on the implementation of the Bayes classification. The network level extends the results obtained at the host level to the rest of the local area network. The approach provides the exchange of the results obtained by the Bayes classification for further use by other program units of the distributed system.

The results of the developed classifier show that representation of the botnets' samples for different classes and subclasses is sufficient for efficient botnet detection. The results of the experiment demonstrated, that the accuracy of the botnet's detection is up to 88%.

7. THE FUTURE WORK

The future work is to develop new methods for botnet detection, which will be focused on the architecture of the distributed systems. It should involve the advantages over the host methods, extending by new botnets samples for more efficient botnet detection.

8. REFERENCES

[1] TrendMicro, 2019, [Online]. Available at: https://www.trendmicro.com/vinfo/us/security/ news/botnets.

[2] ESET Endpoint Security, 2019, [Online]. Available at: https://eset.ua/ua/ products/for_ business/security/endpoint_security.

[3] Dr. Web CureNet, 2019, [Online]. Available at: https://curenet.drweb.ru.

[4] Symantec Endpoint Protection, 2019, [Online]. Available at: https://www.anti-malware.ru/ reviews/Symantec_Endpoint_Protection.

[5] Malwarebytes Endpoint Security, 2019, [Online]. Available at: https://ru.malware bytes. com/business/endpoint security.

[6] Network Admission Control, 2019, [Online]. Available at: https://www.cisco.com/web/RU/ products/hw/wireless/secure/cnac.html.

[7] Kaspersky Administration Kit, 2019, [Online]. Available at: https://support.kaspersky.ru/ learning/courses/kl_102.80/intro/section1.

[8] S. Miller, C. Busby-Earle, "The role of machine learning in botnet detection," *Proceedings of the 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, Barcelona, 2016, pp. 359-364, DOI: 10.1109/ICITST.2016.7856730.

[9] K. Alieyan, A. ALmomani, A. Manasrah, M.M Kadhum "A survey of botnet detection based on DNS," *Neural Computing and Applications*. vol. 28, no. 7, pp. 1541-1558, 2017.

[10] M. Eslahi, W.Z. Abidin, and M.V. Naseri, "Correlation-based HTTP Botnet detection using network communication histogram analysis," *Proceedings of the Application, Information and Network Security (AINS)*, Miri, Malaysia, November 13-14, 2017, pp. 7-12.

[11] A. Pronoza, L. Vitkova, A. Chechulin, I. Kotenko, "Visual analysis of information dissemination channels in social network for protection against inappropriate content," *Proceedings of the 3rd International Scientific Conference Intelligent Information Technologies for Industry*, Sochi, Russia, September 17-21, 2019, vol. 2, pp. 95-105.

[12] M. Komar, A. Sachenko, V. Golovko, V. Dorosh, "Compression of network traffic parameters for detecting cyber attacks based on deep learning," *Proceedings of 2018 IEEE 9th International Conference on Dependable Systems Services and Technologies DESSERT'2018*, Kiev, Ukraine, May 24-27, 2018, pp. 44-47.

[13] S.H. Li, Y.C. Kao, Z.C. Zhang, Y.P. Chuang and D.C. Yen, "A network behavior-based botnet detection mechanism using PSO and K-means," *Journal ACM Transactions on Management Information Systems (TMIS)*, vol. 6, issue 1, pp. 1-30, 2015.

[14] M. Stevanovic and J. M. Pedersen, "An analysis of network traffic classification for botnet detection," *Proceedings of the 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment* London, UK, June 8-9, 2015, pp. 1-8.

[15] M. Sun, G. Xu, J. Zhang, D. Kim, "Tracking you through DNS traffic: Linking user sessions by clustering with Dirichlet mixture model, " in *Proceedings of the 20th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, Miami, FL, US, November 2017, pp. 303-310.

[16] K. Schomp, M. Rabinovich, and M. Allman, "Towards a model of DNS client behavior," *Proceedings of the International Conference on Passive and Active Network Measurement,*

Heraklion, Crete, Greece, 31 March - 1 April, 2016, vol. 9631, pp. 263-275.

[17] J. Zheng, Q. Li, G. Gu, J. Cao, D.K. Yau, J. Wu, "Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, issue 7, pp. 1838-1853, 2018.

[18] M. Kuhrer, T. Hupperich, J. Bushart, C. Rossow and T. Holz. "Going wild: Large-scale classification of open DNS resolvers," *Proceedings of the ACM Internet Measurement Conference (IMC)*, Tokyo, Japan, October 28-30, 2015, pp. 355-368.

[19] N. Koroniotis, N. Moustafa, E. Sitnikova, J. Slay, "Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques," *Proceedings of the International Conference on Mobile Networks and Management*, Springer, Cham, 2017, pp. 30-44.

[20] O. Savenko, S. Lysenko, A. Kryschuk, "Multi-agent based approach of botnet detection in computer systems," in *Communications in Computer and Information Science book series*, vol. 291, 2012, pp. 171-180.

[21] S. Lysenko, O. Savenko, A. Kryshchuk, Y. Klyots, "Botnet detection technique for corporate area network," *Proceedings of the IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems IDAACS'2013*, Berlin, Germany, September 2013, vol. 1, pp. 315-320.

[22] A. Karim, R.B. Salleh, M. Shiraz, et al., "Botnet detection techniques: review, future trends, and issues," *Journal of Zhejiang University SCIENCE C*, vol. 15, pp. 943–983, 2014. https://doi.org/10.1631/jzus.C1300242

[23] T. Sochor, M. Zuzcak, "Attractiveness study of honeypots and honeynets in internet threat detection," *Communications in Computer and Information Science*, vol. 522, pp. 69-81, 2015.

[24] W. Wu, J. Alvarez, C. Liu, H. M. Sun, "Bot detection using unsupervised machine learning," *Microsystem Technologies*, vol. 24, issue 1, pp. 209-217, 2018.

[25] M. Mahmoud, M.P. Nir, A. Matrawy, "A survey on botnet architectures, detection and defences," International Journal of Network Security, vol. 17, issue 3, pp. 264-281, 2014.

[26] Y. Meidan *et al.*, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, DOI: 10.1109/MPRV.2018.03367731.

Oleg Savenko *is a Professor and Dean of the Faculty of Programming and Computer and Telecommunication Systems, Khmelnytsky National University. He earned his B.Eng. Degree in Kamyanets-Podilsky State Pedagogical Institute in 1993 and his PhD Degree in Vinnitsa State Technical University in 1999. His main Areas of Research Interest are Methods for malware detecting, Operating Systems and Artificial Intelligence.*

Anatoliy Sachenko *is Professor and Head of Dept for Information Computer Systems and Control, and Principal Investigator of the Research Institute for Intelligent Computer Systems, Ternopil National Economic University. He earned his PhD Degree in Electrical Engineering at L'viv Physics and Mechanics Institute, Ukrainian National Academy of Science, in 1978 and his Doctor of Technical Sciences Degree in Electrical and Computer Engineering at Leningrad Electrotechnic Institute in 1988.*

His main Areas of Research Interest are Computational Intelligence in Applications, Distributed Measuring Systems and Networks, Intelligent Cyber Security, Wireless Sensor Networks, IT Project Management.

Sergii Lysenko*, is an Associate Professor of the Department of Computer Engineering and System Programming, Khmelnytsky National University. He earned his B.Eng. Degree in Khmelnytsky National University in 2005 and his PhD Degree in Ternopil National Economic University in 2011.*

His main research interests are Self-adaptive detection systems for cyber-threats in computer networks, Methods of detecting cyber attacks in corporate networks and robotics.

***George Markowsky**, currently is a Professor and Chair of Computer Science Missouri University of Science and Technology. George Markowsky has published 115 journal papers, book chapter, book reviews and conference papers on various aspects of Computer Science and Mathematics.*

He has written or edited 15 books and reports on various aspects of computing. He also holds a patent in the area of Universal Hashing. His interests range from pure mathematics to the application of mathematics and computer science to biological problems. He has also built voice controlled and enhanced keyboard terminals for use by paralyzed individuals.

***Nadiya Vasylkiv** is an Associate Professor at the Department of Information Computing Systems and Control, Ternopil National Economic University. She graduated from Ivan Franko Lviv National University with a degree in Physics in 1981. She earned her PhD on temperature measurement in 2011.*

Her main research interests: information system design, IT project management, metrology of data acquisition systems.