Mathematical Problems of Computer Science 58, 67–83, 2022.
doi: 10.51408/1963-0094

UDC 004.725, 004.852

Research of Obfuscated Malware with a Capsule Neural Network

Timur V. Jamgharyan

National Polytechnic University of Armenia
e-mail: t.jamgharyan@yandex.ru

Abstract

The paper presents the results of a research of using transfer training of the capsule neural network to detect malware. The research was carried out on the basis of the source code of malware using the context-triggered piecewise hashing method. The source codes of malware were obtained from public sources of software. Verification of the capsule neural network learning results was carried out using a trained convolutional neural network, and publicly available sources of test to malware. The research was conducted on six types of malware. Software source code, part of capsule neural network training datasets, pre-trained capsule neural network, and full research are publicly available at https://github.com/T-JN

Keywords: Capsule neural network, Context triggered piecewise hashing, Edit distance, Intrusion detection system, Transfer learning.
Article info: Received 9 June 2022; accepted 24 November 2022.

1. Introduction

Malware injected into Infrastructure through zero-day vulnerabilities in network equipment is a huge cybersecurity problem. The network infrastructure (NI) protection architecture implies the construction of a multi-level, complementary security system. Part of the NI security design is an intrusion detection system (IDS).

In the studies [1]-[5], the types of IDS, the ways of their application and the mechanisms of their work are considered in detail. «Classic» IDS can be classified as:

- host-based IDS, that is detection of attacks on a specific network node,
- network-based IDS, that is, detecting attacks on the network or its segment.

Existing IDS that do not use machine learning (ML) in their functionality (both proprietary and open source) [6]-[9], have one common drawback: they all respond to the threat that is embedded in the rule sets. There is also a high probability of various false positives: (true positive, true negative, false positive, false negative) [10]. Malware is the most common threat

vector in most operating environments [11]. The IDS software ecosystem offers many utilities and application suites that can help collect signals from all types of network traffic [12].
For IDS operating without the use of ML at different levels of the Open System Interconnection (OSI) model [13], the task of detecting malware modifications was secondary. Basically, the task of detecting and neutralizing malware was assigned to antivirus software. But with the convergence of attacks at different levels of the OSI model and the emergence of software-defined networks (SDN), new types of threats and possible attacks arise, the neutralization of which by «standard» methods is difficult [14]-[15]. New systematic approaches are required to solve these problems. With the increase in the growth of attacks built on the basis of ML and machine-to-machine (M2M), new threats to the NI also arise. The requirements for security systems are increasing. The convergence of system, network and cloud services increases both the «attack surface» [16] and the «attack space power» [17]. Of particular danger are attacks «designed» using ML [18]-[20]. Researchers are working on the application of ML to create and build a new type of IDS [21-25]. Unlike «classic» IDS, built on the basis of ML can be further trained, being in one way or another a malware generator [26]-[28]. At this stage, both conceptually new solutions in the field of ML application in IDS are being developed, as well as improvements to existing ones. The papers [29]-[32] consider the issues of using ML to create one or another type of IDS. Researchers and developers of ML-based IDS are faced with a large number of tasks that need to be solved, due to the novelty of this area of information security.

- The task of having annotated data for training a neural network (Annotation is the process of labeling raw data so that it can become training for machine learning [11]). No algorithm can handle really bad data. There are many different requirements for training datasets, in particular, representativeness and «noiselessness». [33]. Unlike neural networks that process images, sound, text, etc., for which there are verified datasets [34]-[39], datasets for training an IDS must to some extent, consist of malware. Researchers have access to certain resources that supply research malware [40]-[46], but these resources make them public with a delay.
- The task of increasing the learning rate of IDS built on the basis of ML. Unlike other neural networks where the main attention is paid to the quantity and quality of training data, in intrusion detection systems built on the basis of ML, in many cases, the speed of learning is also important. As shown in [47], since the emerging malware not included in any database has a different data distribution compared to the original training samples, the efficiency of model detection will decrease when it encounters new malware.
- The task of correctly calculating the degree of threat in an attack using ML [48]. When developing an IDS based on ML, it is necessary to correctly calculate the degree of threat to the protected NI.

In addition to general tasks, there are also specific tasks: since each group and type of malware requires its own specific detection methods [49]-[50].

- Detection based on signature analysis, where a database of malware hashes is used as a signature,
- Detection based on Indicator of Compromise (IoC). It is a set of artifacts based on which malware can be detected: registry branches, loadable libraries, IP addresses, byte sequences, software versions, date and time triggers, ports involved [51].
- Research based on context triggered piecewise hashing (CTPH), (context triggered piecewise hashing is a method of calculating piecewise hashes from input data [52]). Malware developers use various techniques to change the original malware signature to make hashes harder to detect: encryption, obfuscation, reordering of files and libraries, re-distribution and code building in order to fool the detection system, giving malware a

new look and changing the hash values. In this case, malware remains undetected for some time [53].

Various researchers are considering the use of CTPH techniques for malware detection. In [54], the issue of applying transfer learning to solve the problem of malware domain bias is considered, and in [55], the issue of automatic malware family identification and classification through online clustering is considered. But the main issues of preparing malware datasets and training IDS based on ML remain open.The issue of increasing the performance of an IDS based on ML with a small set of training datasets remains relevant. In this paper, a method for applying transfer learning of a capsule neural network with the calculation of CTPH and editing distance to increase the learning rate and detection of malware is investigated. The Levenshtein method [56] (Equation 1) and the method using the *ssdeep* program [57] were chosen as the mathematical apparatus for calculating the editorial distance. To assess the quality of binary learning, the Matthews correlation (Equation 2) [58] was used. The source codes of the malware for creating a set of annotated datasets were taken from open sources. The following malware was used: *mimikatz, athena, engrat, grum, surtr, dyre.*

$$D(i,j) = \begin{cases} 0, & i = 0,\ j = 0 \\ i, & j = 0,\ i > 0 \\ j, & i = 0,\ j > 0 \\ \min\{ & \\ \quad D(i,j-1) + 1, & j > 0,\ i > 0 \\ \quad D(i-1,j) + 1 + m(M[i], N[j]), & \\ \} & \end{cases} \tag{1}$$

Levenshtein editorial distance calculation equation,

where, D - the editorial distance, M, N- the length of strings obtained as a result of CTPH over some alphabet (in this case HEX), i - remove step from the first line, j-insert into the first line.

$$\phi = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \tag{2}$$

where,

ϕ - Matthews correlation

TP - true positive,

TN -true negative,

FP -false positive,

FN - false negative.

A capsule neural network was chosen as a transfer learning model. The choice of the capsule network is due to the following reasons:

- the capsule network does not require a large amount of training data, which is critical for this research,
- the capsule network explores hierarchical relationships, which allows detecting possibly probable versions, in the presence of a primary code (a fragment of the main code) of malware,
- the capsule network allows searching even in obfuscated source code with a minimum malware representativeness value,

- the capsule network is the most easily adaptable to changing the learning algorithm compared to other neural networks.

2. Diagrams of Neural Networks

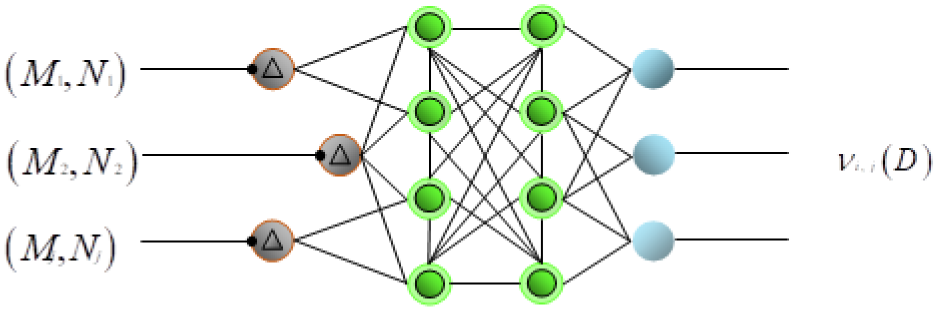
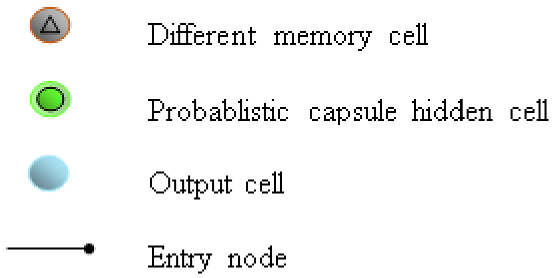


Fig.1. Diagram of a capsule neural network.



The nonlinearity function of the capsule network is determined by (Equation 3) [59].

$$\boldsymbol{v_i} = \frac{||s_i||^2}{1+||s_i||^2}\frac{s_i}{||s_i||}, \qquad (3)$$

where, s_j- the result obtained in the previous step, $\boldsymbol{v_i}$ - the result obtained after applying the non-linearity. The left side of the equation performs additional compression, and the right side of the equation performs unity scaling of the output vector.

The trained convolutional neural network (Fig. 2) was chosen as a test to check the reliability of the output data. As «weight coefficients» of the convolutional neural network, the value of CTPH was calculated the used *ssdeep* software.

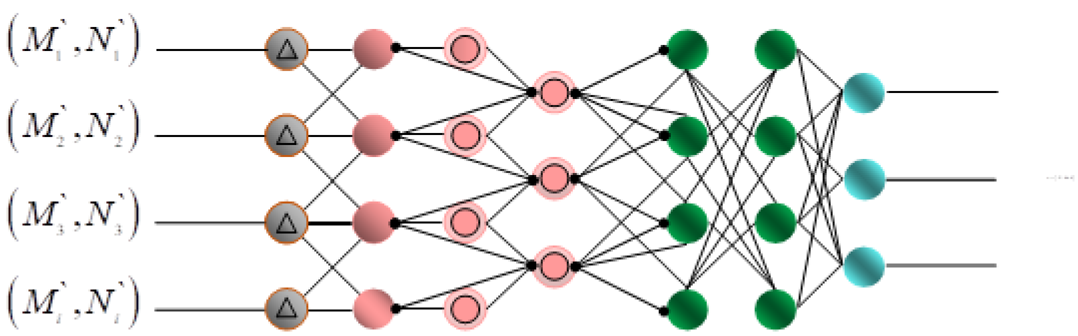
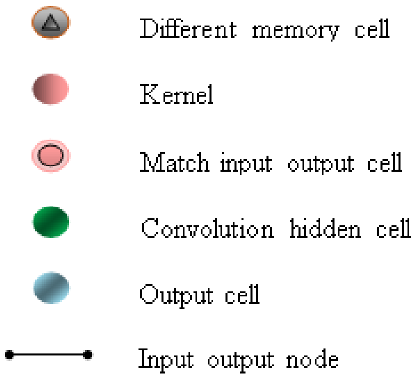


Fig. 2. Diagram of a convolutional neural network.



Verification of the results obtained from both neural networks was carried out using public malware detection services [60]-[61]. The developed software algorithm is shown in Fig.3.

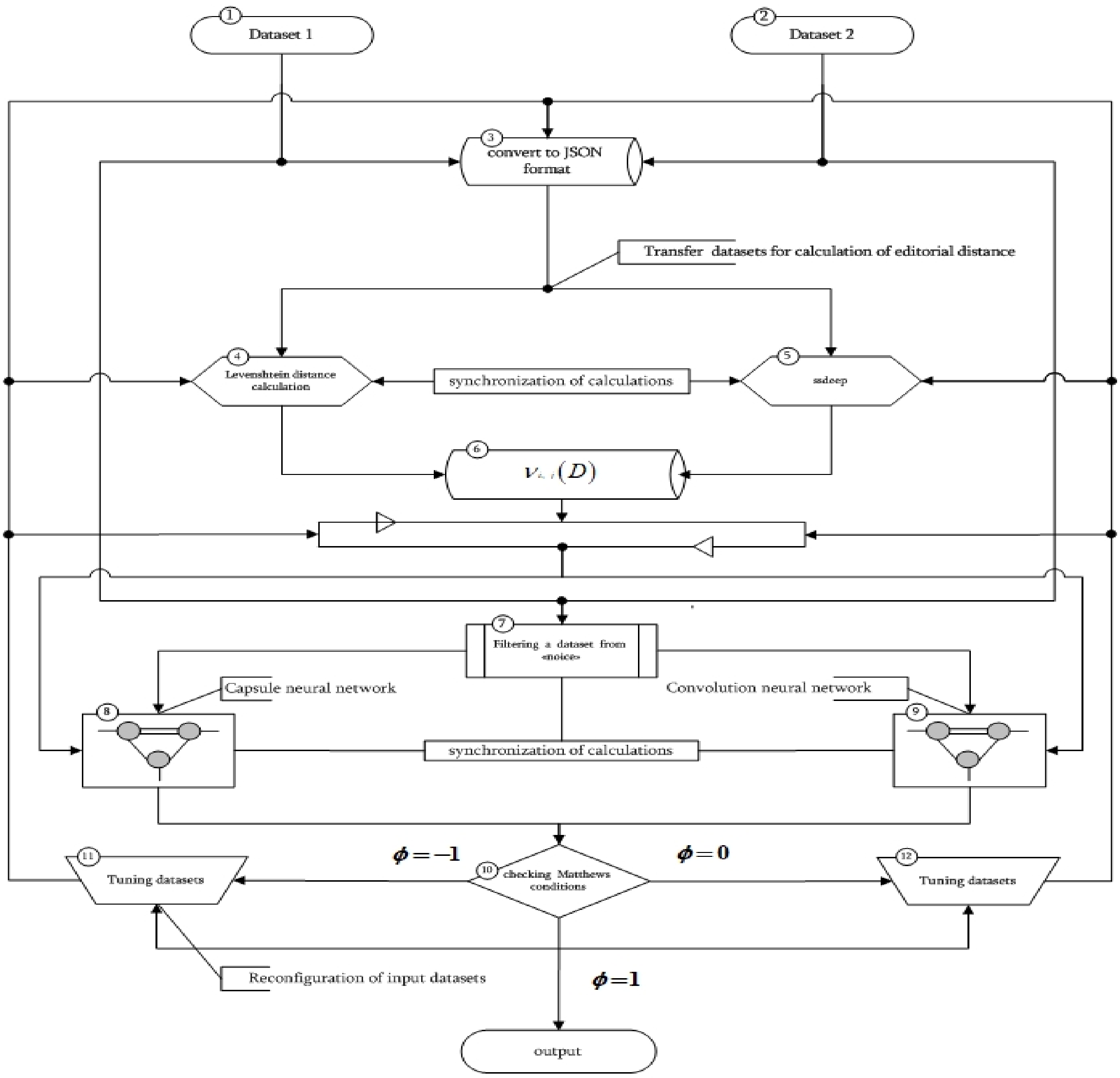


Fig. 3. Algorithm of the developed software.

Algorithm operation:

Operations on the input data of the research.

- The dataset generated from the malware source code was obfuscated using various tools [62]-[63] and prepared for training a capsule neural network (dataset 1).
- The same non-obfuscated dataset (dataset 2) generated from the malware source code was prepared to train a convolutional neural network.

A total of 1000 annotated datasets of various sizes (20.40, 80, 128, 256, 512, 1024 bytes) were prepared for *mimikatz, athena, engrat, grum, surtr, dyre* software.

Steps 1, 2: input of the initial malware dataset into the trained neural networks and the conversion module,

Step 3: converting the source dataset to javascript object notation (JSON) format and setting the CTPH step size,

Step 4: calculation of the edit distance by the Levenshtein method,

Step 5: computation CTPH using *ssdeep* software**,**

Step 6: comparison of the values calculated by the Levenshtein method and using the *ssdeep* software,

Step 7:filtering the training datasets of neural networks from «noise» (the full implementation of this part of the algorithm is presented in [33]),

Step 8: training capsular neural network,

Step 9 training convolutional neural network,

Step 10 compute the Matthews correlation and resize the training datasets**.**

- $\phi = -1$ the received output data of both neural networks go beyond the value tolerance
- $\phi = 1$ the resulting outputs of both neural networks are correct (within the permissible deviation value)
- $\phi = 0$ the resulting output of both neural networks is random

Steps 11, 12: reconfiguring the training datasets and resizinge the CTPH.

Table 1 presents the results of calculating the value of CTPH and the editorial distance between the hashes of the obfuscated source code of mimikatz software using capsular, convolutional neural networks, as well as ssdeep software.

Table 2 shows the results of calculating the value of the context-piecewise hash of the obfuscated compiled source code and the editorial distance between the hashes of the mimikatz software using capsular, convolutional neural networks, and also the *ssdeep* software.

In the research, datasets used a comparison between files 20-40, 20-80, 20-128, 20-256, 20-512, 20-1024 bytes, as well as combinations of 40-512, 40-1024, 128-512, 128 -1024 bytes for *mimikatz, athena, engrat, grum, surtr, dyre* malware.

3. Results

Table 1.The results of computing the value of CTPH and the editorial distance between the hashes of the obfuscated source code of mimikatz software

File number in the dataset	mimikatz file hash values (20 byte)	mimikatz file hash values (512 byte)	Editorial distance	Percentage of malware samples calculated using scdeep	Percentage of malware samples computed using convolutional neural networks			Percentage of malware samples computed using capsule neural network		
					Training epoch			Training epoch		
					I	II	III	I	II	III
1.	b9be58b87140f922969c905236829d2436c34400	ef73afe0b3862206e112400dc97a6920c1240ca2	36	10	4	2	9	8	19	39
2.	e1077e747c9486dce1bfda820c078fe300a901fb	081cdfaf631a003a5a5dfa678b52af5c0eb2cbd3	36	13	6	8	7	16	27	42
3.	d86c9ca3861e333dc3376fc5565943551389edd6	72840526d3cecbba084eef91aed9c52cd94855d5	35	25	18	9	9	24	52	78
4.	bd72fda18edc004d5181b57e48a757ac2ed94444	783e9520a25faca4f8152dfc092d7d67e359c5f6	35	28	21	22	24	8	10	12
5.	8ab1d3267a46f953c73b4154b1a261a8e02493d8	ad523321e582956d7b51e9f4bc3763d9305231dc	30	11	3	7	3	34	54	82
6.	dc990c540fc50debf0cdc178101ab107acaef9fe	f2ba969ed8f8ecc7ce57c54c39de5333cf0d6a8e	36	23	11	16	21	16	28	65
7.	b137df3d2083c226f985c0494a9cef753034ac6d	f7fd9ed34bc6ead485bd5e7c1b9f9f13f30fddba	34	13	10	9	12	16	27	46
8.	9efa06fa6567be9554db5c351da39c9c084306e0	f7fd9ed34bc6ead485bd5e7c1b9f9f13f30fddba	33	21	15	15	17	31	46	79
9.	4f5ec65628d2bde662a408854a41caea98c0f44f	f5cd09b85a44df103b21ea9c4d02c564fcb19191	35	64	32	30	42	38	37	48
10.	5329b04a348368967844f421453563001ad4ab89	37a56e3a4acbef5420994c0d7864125e53f5aaa3	36	22	8	11	16	27	48	61
11.	95a56dfdfd7c8550afb8ab2474916bb63e58bb15	37a56e3a4acbef5420994c0d7864125e53f5aaa3	33	16	12	13	15	27	41	68
12.	aececb9dccd29fd5dd9c0559ad62afb84af374b2	51168e0c2ab45361cf05834a721cd4aba48098be	34	19	11	12	18	36	49	73
13.	14791ec8ec19ca534367c54f008b8439eea89f09	497a16d6dd757f05fb884994c71bea880e87ad18	35	11	18	29	25	37	49	68
14.	dbfb0b8c0a28ea8bade6306f9e8589ee1c310a39	c6ca0e98e0a66c45838fb254aec474553850ab91	34	16	14	21	29	52	58	71
15.	c91e176518b7e42450e2c28d45bf31a1b3178240	7ad0cc0f4ba8c767fac7f0a4f7ec192b3a60ec9e	36	18	16	19	28	29	43	68
16.	04b66940a08ac7adb0cdf19382a8169d0c256c09	5db88a72cdcfe90ff9871eae5bf8d2b617d73b0a	37	26	11	19	36	39	56	73
17.	67b4a269a360b994d7769e4b40220c8b59c219b0	fa926a049a1d9d72126bd07f1a1b87326b5e355b	34	41	27	11	29	26	58	61
18.	c2cdacd22e871ecef12b0cbc8caf4559eecfa084	817c64fed50532e58dd21a8812c65fe10a250bd0	36	16	15	16	26	31	46	74
19.	4202fc70b1301ec50b1f64ca525de6d31825787d	38bc177d79492834356f1cce4f9120599f41e952	36	18	17	19	21	28	37	49
20.	20b5c47533cb97d72f90895ea1ffe27695063e54	818b59add29456248836864d46c146d9d930d8a2	37	19	8	16	34	24	37	58

In training epochs 1-3, the results of the capsular neural network are better than the results of the convolutional neural network and ssdeep software, except for file №4 in the dataset, which is included in the statistical error.

Table 2. The results of calculating the value of CTPH and editorial distance between hashes of the compiled mimikatz source code.

File number in the dataset	mimikatz file hash values (20 byte)	mimikatz file hash values (512 byte)	Editorial distance	Percentage of malware samples calculated using ssdeep	Percentage of malware samples computed using convolutional neural networks			Percentage of malware samples computed using capsular neural network		
					Training epoch			Training epoch		
					I	II	III	I	II	III
1.	d7e4e9abedd0949b8bcf f30c7abbdad97b182be8	51f028f6b078f51583e0 a048d9bc577b6a4e17b9	37	25	23	31	42	17	19	23
2.	2c0e9d614fab60e18bd4 2e99659974a3d298a9ae	7f966e5a707dd69c13b5 de45c9765a9be437e642	35	16	18	14	22	8	11	9
3.	f76606cb6fae082991eb 271af5ab7629d592cb04	fb96549631c835eb239c d614cc6b5cb7d295121a	32	28	27	36	45	16	17	14
4.	14da593832768f0a08e8 ecd46363936eef096dcc	72ac7a00a3c2a0a825cd 016d71b0d587c6cc3f46	36	23	16	22	34	18	20	16
5.	7f01a23afa1bcecdfdbb 25b953c4f15366eaba51	35139ef894b28b73bea0 22755166a23933c7d9cb	37	37	34	41	48	27	29	23
6.	1ca12a53c82cdd508054 bdcdbe5256ccdd44c13c	918b1c05e576f4b90fce 15a06bc3442d72852a3c	35	48	44	53	61	34	31	28
7.	a7f0499bf3eb6180d4da 748426822404e46dea13	4759f2ba1ba20f493664 dbf5e36c1a1ec0d75658	36	15	11	13	12	8	2	3
8.	aec2a4accb7ca456a57a c4426e8f51c2e6a8b143	902a2d132f213700b5de fbefe7567f68ca8e234a	35	19	18	26	29	16	10	13
9.	582d2ceff8f4f493f3a9 d45c71286255946a7d37	b2fd9a1405ba74fc360e 1784961176b2b88bf5c9	37	39	28	48	57	25	23	12
10.	a25a87930b155282e138 35142ad63cea1994d02d	c47419fdd4d6f146e430 64b9ddb859a250404500	36	53	47	40	57	34	29	47
11.	2f7b14912dddcf7c1c7a ebb49955cb5bf0ab3257	b521d7652866027a7e5b 43c6269d7c81ffb5a86e	36	28	30	37	44	14	19	23
12.	fd5fd2f7953cf5630f74 c2933b378d4381367ddd	9de4bfa1fdb6c90637d3 5492ec14ee10a3967997	33	56	49	53	67	42	48	34
13.	e88dac72cd8ac64360d9 5fb15e8ea9aaa8794f8c	1eb796fd1ff7dda036fc a37d0f31aab19dedab1a	37	24	29	48	52	17	23	15
14.	efa91cc773ee2c32ba51 2ffce8db8a3760bda564	99828f68be57c53ff954 5f79e32bdb36050bf93b	32	19	27	29	37	13	18	28
15.	f9980d6122acf1bf54a6 8e49d15507fbc3ce7c1f	2400b40333821b00b5d0 b67f20f5f0e30ebf02dd	36	56	44	58	63	37	34	39
16.	c5d4d95ce32029e1150a 20d2f836b7b2c6e49546	dfb380d8b0709104c606 978092c7164160f32887	37	29	27	35	38	21	17	34
17.	5156507d0b07bd9eaafe 56815e1a04a0eaa1a8e9	bd951f174a8f0f211c62 bc1869d69f581788ee59	37	48	27	44	56	25	38	14
18.	14fd3fa5756432336c73 656c76f4751aa6f707f9	b9acd4446a9ee133799f a3d8f3e35e001c616776	37	16	24	36	38	8	10	11
19.	f1d8238c9141f46246bf 2193908b1be6f87b09f8	f1513655d577bf56bcf86 2b1851e66bb683d373c	33	56	48	56	61	32	27	46
20.	50effcaad368f00bfc71 2105a708ff917f9f95d0	49a48ed249c7b82959aa 85b9470938bbcc9c45cc	36	36	27	38	46	16	28	31

In epochs 1-3 of training for compiled software, the results of the capsule neural network are worse worse than the results of the convolutional neural network and ssdeep software.

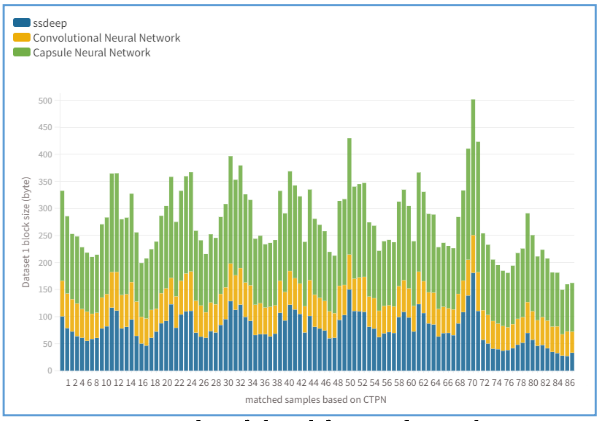


Fig. 4. CTPH results of the obfuscated mimikatz source code.

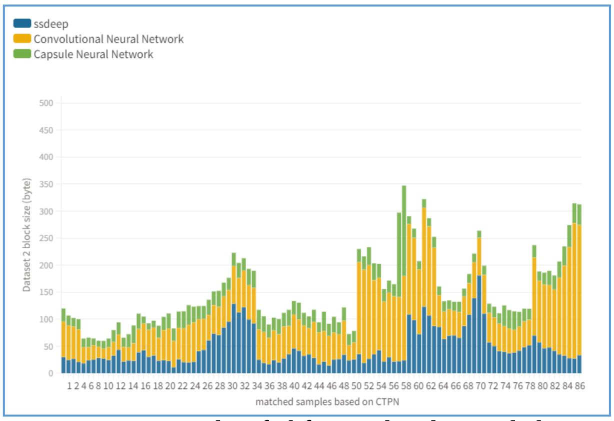


Fig. 5. CTPH results of obfuscated and compiled mimikatz source code.

The use of a convolutional neural network is not always justified, since the degree of detection is comparable to the degree of detection by *ssdeep* software. The use of a capsule neural network for malware detection is justified in the presence of the source code (even in an obfuscated state), since even after the first training epoch, the detection results are not worse (and in most cases better) than the detection results using ssdeep and a trained convolutional neural network.

Tables 3 and 4 present the results of the studies of the operation of capsule and convolutional neural networks, based on datasets obtained from the *obfuscated mimikatz source code* with three training epochs and a variable block size of CTPH.

Table 3. Number of detected threats.

Number of datasets (dataset 1)	Number of datasets (dataset 2)	The number of samples detected and classified as threats on different sizes (20, 40, 128 bytes) and three epochs (I, II, III) of training by a capsule neural network									The number of samples detected and classified as a threat at different sizes (20, 40, 128 bytes) and three epochs (I, II, III) of training by a convolutional neural network									Number of detected but mismatched malware samples *		
CTPN size (byte)		20			40			128			20			40			128					
Training epoch		I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III
100	100	7	7	9	11	13	12	12	15	18	3	3	4	4	6	6	9	10	1	-	-	1
200	200	10	11	11	12	14	16	17	17	21	5	4	6	6	8	5	8	5	6	-	1	2
300	300	12	12	14	16	18	23	28	29	22	8	7	8	8	9	11	13	15	16	1	1	2
350	350	12	13	15	15	16	18	21	26	25	7	7	11	10	12	18	16	18	19	2	2	3
450	450	14	16	19	19	22	26	29	34	38	10	9	11	12	16	18	18	21	20	2	1	4
500	500	14	16	18	19	21	27	29	33	36	11	10	13	16	15	15	17	19	19	2	2	4
600	600	22	25	29	30	34	35	39	41	44	14	15	11	19	24	26	20	25	26	3	3	3
800	800	37	41	46	48	52	55	57	57	60	22	26	27	29	34	37	39	44	45	5	4	6
950	950	42	42	46	47	58	60	66	68	68	28	29	28	31	33	39	42	46	49	4	4	4
1000	1000	42	43	47	50	51	59	61	65	69	34	33	35	30	35	39	49	52	55	5	6	3

*The number of detected but mismatched malware samples separately detected by both neural networks. These samples were output to a special dataset and verified by publicly available malware detection resources.

Table 4. Number of detected threats.

Number of datasets (dataset 1)	Number of datasets (dataset 2)	The number of samples detected and classified as threats at different sizes (256, 512, 1024 bytes) and three epochs (I, II, III) of training by a capsular neural network									The number of samples detected and classified as a threat at different sizes (20, 40, 128 bytes) and three epochs (I, II, III) of training by a convolutional neural network									Number of detected but mismatched malware samples *		
CTPN size (byte)		256			512			1024			256			512			1024					
Training epoch		I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III
100	100	18	14	16	14	16	19	8	12	14	7	11	14	9	11	14	7	8	11	-	1	1
200	200	18	12	12	14	18	19	11	13	10	3	4	3	5	8	11	5	9	14	1	1	2
300	300	17	19	16	14	17	12	10	21	23	9	11	10	8	12	9	8	8	13	-	2	2
350	350	18	18	21	18	21	23	23	27	27	9	15	17	12	18	14	14	11	12	2	2	3
450	450	22	26	28	29	29	34	20	23	25	12	15	13	20	16	16	17	29	13	2	5	3
500	500	23	24	29	31	33	30	28	21	32	16	12	15	22	22	25	28	26	25	3	7	7
600	600	28	31	30	32	35	39	34	38	41	20	24	21	24	28	25	29	34	31	5	6	6
800	800	37	37	39	41	46	39	42	46	49	31	28	34	34	25	27	39	32	34	7	9	11
950	950	48	53	53	52	58	56	64	65	56	34	30	31	35	38	38	39	42	45	11	9	10
1000	1000	47	52	51	56	61	60	64	66	68	40	42	46	42	44	44	47	49	51	8	11	12

Fig. 6 shows a report from the *virustotal* service when examining one of the *mimikatz* malware samples detected by neural networks. In particular, the virustotal service did not detect either the file type or whether CTPH (based on ssdeep) belongs to a particular type of malware.

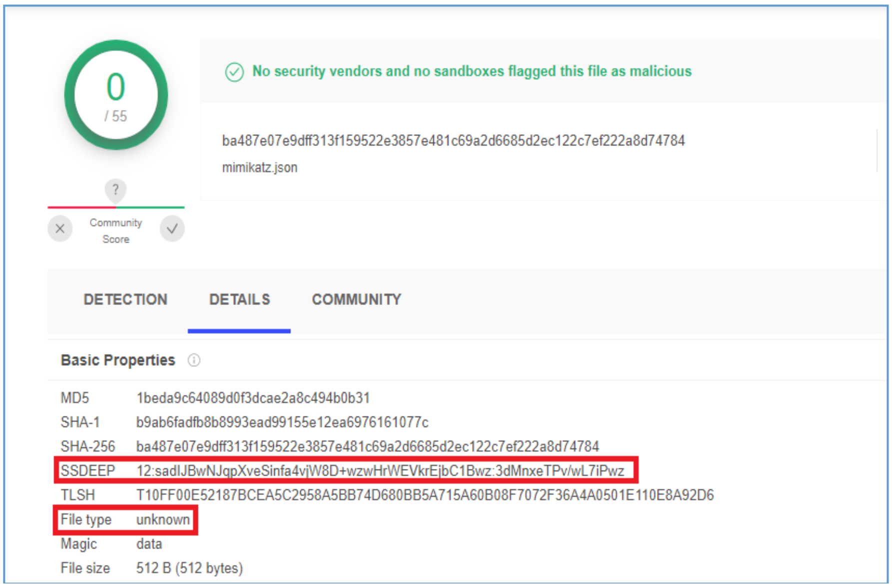


Fig. 6.Virustotal service report.

Tables 5 and 6 present the results of the studies of the operation of capsule and convolutional neural networks, based on data sets from the *obfuscated compiled code* of the *mimikatz* software.

Table 5. Number of detected threats.

Number of datasets (dataset 1)	Number of datasets (dataset 2)	The number of samples detected and classified as threats at different sizes (20, 40, 128 bytes) and three epochs (I, II, III) of training by a capsular neural network									The number of samples detected and classified as a threat at different sizes (20, 40, 128 bytes) and three epochs (I, II, III) of training by a convolutional neural network									Number of detected but mismatched malware samples *		
CTPN size (byte)		20			40			128			20			40			128					
Training epoch		I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III
100	100	2	1	2	3	2	3	3	4	4	2	2	3	3	3	4	5	2	3	-	-	-
200	200	3	2	3	3	4	2	2	3	3	1	1	2	2	3	2	4	3	4	-	-	-
300	300	3	4	4	4	4	5	3	5	5	2	3	3	4	3	4	4	4	4	-	-	1
350	350	3	3	4	4	5	5	5	6	6	3	3	3	3	4	5	5	4	4	-	1	1
450	450	4	5	5	5	6	6	6	8	9	3	4	4	4	5	6	5	7	7	-	1	-
500	500	3	5	5	5	6	8	8	9	11	4	4	5	5	7	9	9	10	10	-	2	2
600	600	5	6	6	6	8	9	11	11	12	5	4	7	7	9	11	10	11	10	1	1	1
800	800	7	6	7	7	8	11	13	14	14	6	8	9	8	8	9	8	11	13	2	1	2
950	950	9	9	10	11	9	11	12	15	15	8	10	10	11	13	15	14	15	17	2	2	3
1000	1000	11	13	14	14	14	15	17	19	18	10	11	11	11	13	16	18	21	23	2	4	4

Table 6. Number of detected threats

Number of datasets (dataset 1)	Number of datasets (dataset 2)	The number of samples detected and classified as threats at different sizes (256, 512, 1024 bytes) and three epochs (I, II, III) of training by a capsular neural network									The number of samples detected and classified as a threat at different sizes (256, 512, 1024 bytes) and three epochs (I, II, III) of training by a convolutional neural network									Number of detected but mismatched malware samples *		
CTPN size (byte)		256			512			1024			256			512			1024					
Training epoch		I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III
100	100	9	11	12	12	14	14	15	16	16	8	8	10	11	13	12	11	11	10	-	-	1
200	200	10	12	13	14	13	13	15	15	12	11	10	11	12	11	13	12	13	14	-	1	1
300	300	11	12	12	15	17	18	19	18	18	10	12	13	12	14	14	15	14	14	-	-	-
350	350	11	11	12	12	12	16	15	11	14	14	12	13	15	15	15	18	19	21	-	1	2
450	450	13	12	13	13	15	15	16	17	18	11	12	13	14	16	16	15	17	19	2	3	3
500	500	12	14	14	14	15	14	15	11	12	11	10	11	13	14	12	15	15	16	-	1	2
600	600	10	11	12	10	12	12	12	14	13	9	10	11	12	10	10	14	15	14	1	2	2
800	800	12	14	15	15	16	17	17	18	18	16	14	15	15	16	17	18	21	19	2	3	3
950	950	12	13	12	14	15	15	16	18	19	12	12	13	14	15	16	12	15	16	2	3	4
1000	1000	12	12	13	13	15	16	16	17	18	11	10	12	15	16	17	18	19	20	2	2	3

Given the malware source code (or fragment), the capsule neural network performs better than the convolutional neural network in detecting obfuscated malware. But when compiled, the detection performance of the capsular neural network decreases. Also, both neural networks separately detected a small set of data and software fragments classified as malware. Figures. [7]-[12] show a visualization of the output data of a capsule neural network with 3 training epochs and CTPN datasets, 20, 40, 80, 128, 256, 512 bytes.

Fig. 7. Visualization of malware detection results by capsule neural network. (I training epoch, CTPH size 20 bytes)

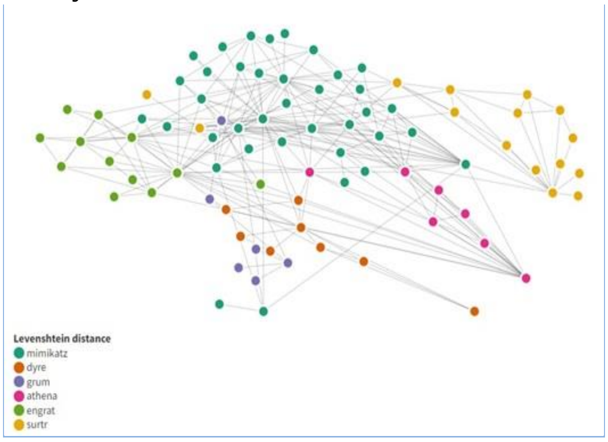


Fig. 8. Visualization of malware detection by capsule neural network. (I training epoch, CTPH size 40 bytes)

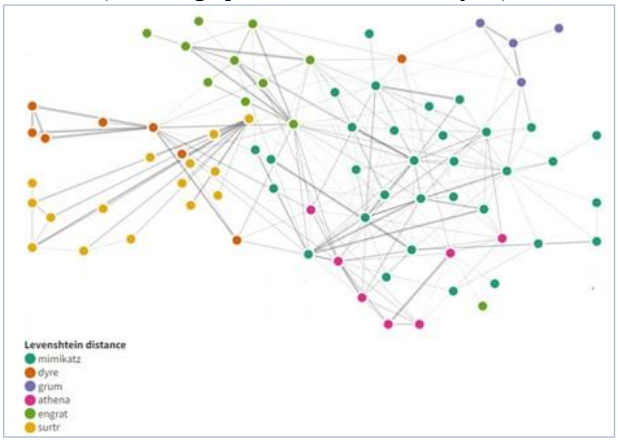


Fig. 9. Visualization of malware detection results by capsule neural network. (II training epoch, CTPH size 80 bytes)

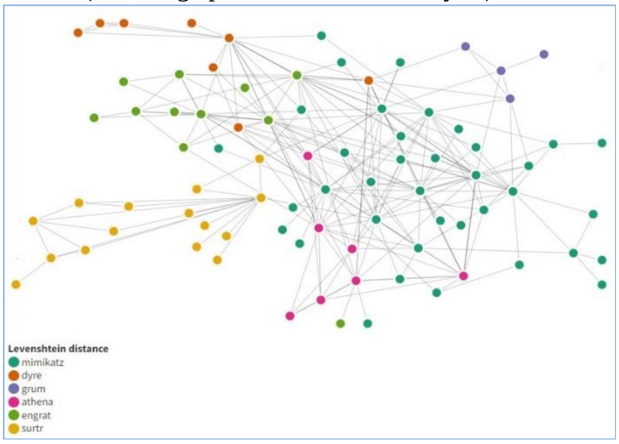


Fig. 10. Visualization of malware detection by capsule neural network. (II training epoch, CTPH size 128 bytes)

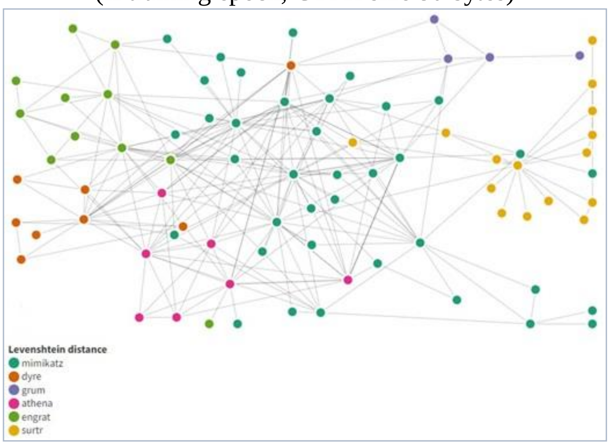


Fig. 11. Visualization of malware detection results by capsule neural network. (III training epoch, CTPH size 256 bytes)

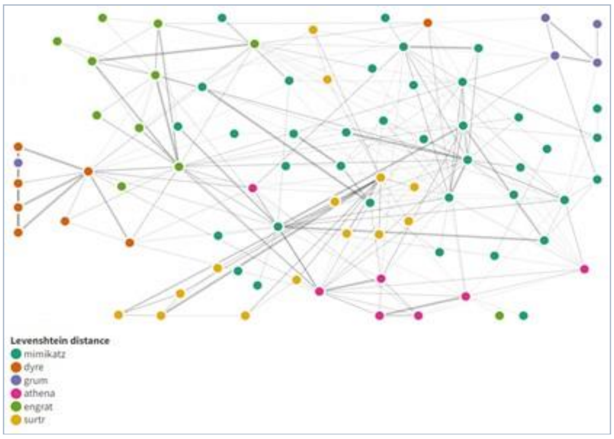


Fig. 12. Visualization of malware detection results by capsule neural network. (III training epoch, CTPH size 512 bytes)

With an increase in the size of the CTPH files (interval 256, 512, 1024 bytes) for training the capsule network, the increase in the detection of the number of malware code fragments is insignificant (0.3-0.5%, Fig. 7, Fig. 8, Table 6) in contrast to files 20 , 40, 128 bytes (12-14% increase). But increasing the size of the CTPH file allows increasing the editorial distance (Figure 9-12) to granularly group malware by type.

4. Conclusion

This paper proposes the use of transfer learning of a capsule neural network to detect obfuscated malware. Convolutional and capsule neural networks were trained on the same datasets. The source codes of *mimikatz, athena, engrat, grum, surtr, dyre* malware were used as datasets. When building an intrusion detection system using neural networks, their complex application is necessary. Annotated malware datasets are critical when training neural networks. The use of transfer learning of a capsule neural network to detect malware is justified if the source code of the malware or its fragments (preferably the first versions) is available. In this case, the neural network detects malware, even with its high degree of obfuscation. But in the absence of source code, the effectiveness drops, yielding to «standard» means of detecting malware. The use of the CTPH method for generating «weight» coefficients of a neural network is most effective with a small file size of CTPH.
Increasing the editorial distance increases the selectivity of detecting different types of malware.

References

[1] D. Ashok Kumar and S. R.Venugopalan, "Intrusion Detection Systems: A Review" *International Journal of Advanced Research in Computer Science*, vol. 8, no 8, pp.356--370, 2017.

[2] O. Shelukhin, D. Sakalema and A.Filinov, *Detection of intrusions into computer networks.* Hot line-Telecom, 2018.

[3] S. Survey and D. Usha, "A survey of intrusion detection system in IoT devices", *International Journal of Advanced Research (IJAR),* vol 6, pp. 23-31, 2018.

[4] H.Hindy et al., "A taxonomy of network threats and the effect of current datasets on intrusion detection system", *arXiv preprint arXiv:1806.03517,* 2020.

[5] Tuan-Hong Chua and Iftektar Salam, "Evaluation of machine learning algorithms in network-based intrusion detection system", *arXiv preprint arXiv:2203.05232,* 2022.

[6] Snort intrusion detection and prevention system official website. [Online]. Available https://www.snort.org/

[7] Suricata intrusion detection and prevention system official website. [Online]. Available https://suricata.io/

[8] Zeek an open source Network Security Monitoring tool system official website. [Online]. Available https://zeek.org/

[9] Cisco NGIPS system web pages. [Online]. Available https://www.cisco.com/c/ru_ru/products/security/ngips/index.html

[10] F.Maymi and S.Harrris, *CISSP, Exam Guide*, Ninth Edition, Mc Graw Hill, New York, San Prancisco, Singapore, Sydney, Toronto, 2022.

[11] C. Chio and D. Freeman, *Machine Learning and Security,* O`Reilly® , Boston•Sebastopol• Tokyo, 2020.

[12] M. Collins, *Network Security. Through Data Analysis,* O`Reilly® (DMK press), 2020.
[13] ISO/IEC 7498-1, Second edition 1994-11-15. Corrected and reprinted, 1996.
[14] MITRE ATT&CK® official website. [Online]. Available https://attack.mitre.org/matrices/enterprise/
[15] CVE cybersecurity web pages. [Online]. Available https://cve.mitre.org/index.html
[16] OWASP Cheat Sheet Series. [Online].Available https://cheatsheetseries.owasp.org/cheatsheets/Attack_Surface_Analysis_Cheat_Sheet.html
[17] A. Cheremushkin, *"Cryptographig protocols: Main properties and vulnerabilites*", PDM , vol.2 appendix, pp.115-150, 2009.
[18] T. V. Jamgharyan and V.H.Ispiryan, "Model of generative network attack" *Proceedings of 13th International Conference on Computer Science and Information Technologies (CSIT),* Yerevan, Armenia, pp. 90-94, 2021.
[19] A. Ul Haq et al, "Addressing tactic volatility in self-adaptive systems using evolved recurrent neural networks and uncertainty reductions tactics", *arXiv preprint arXiv:2204.10308v1,* 2022.
[20] S. Das, "FGAN: Federated generative adversarial networks for anomaly detection in network traffic", *arXiv preprint arXiv:2203.11106v1,* 2022.
[21] Sk.Tanzir Mehedi, "Dependable intrusion detection system for iot: a deep transfer learning –based approach", *arXiv preprint arXiv:2204.0483v1,*2022.
[22] I. Panagiotis et al, "Securing the Smart Grid: A Comprehensive Compilation of Intrusion Detection and Prevention Systems", *DOI 10.1109/Access,* 2017.
[23] A. S. Dina et al , "Effect of balancing data using synhthetic data on the performance machine learning classifiers for intrusion detection in computer networks", *arXiv preprint arXiv:2204.00144v1,*2022.
[24] T.Nathuya and G.Suseendram, *An Effective Hybrid Intrusion Detection System for Use in Security Monitoring in the Virtual Network Layer of Cloud Computing Technology,* Springer Nature, Singapore, 2019.
[25] E.Pelofske, "A robust cubersecurity topic classification tool", *International Journal of Network Security & Its Application (IJNSA),* vol.14, № 1, pp. 1-25, 2022.
[26] G.Renjith et al, "GANG-MAM: GAN based enGine for modifying android malware" *arXiv preprint arXiv: 2109.13297,* 2021.
[27] F.Zhong et al, "MalFox: Camouflaged adversarial malware example generation based on Conv-GANs againist black—box detectors", *arXiv preprint arXiv: 2011.01509,* 2021.
[28] B.E.Zolbayar et al, "Generating practical adversarial network traffic flows using NIDSGAN", *arXiv preprint arXiv: 2203.06694v1,* 2022.
[29] Md.Ariful Haqua, R.Palit, " A review on deep neural network for computer network traffic classification", *arXiv preprint arXiv: 2205.10830v1,* 2022.
[30] D. Kus et al, "A false sense of security? Revisting the state of machine learning-based industrial intrusion system", *arXiv preprint arXiv: 2205.09199v1,* 2022.
[31] S.Layeghy and M. Portmann, "On generalisibility of mashnine learning-based network intrusion detection systems", *arXiv preprint arXiv: 2205.041112v1,*2022.
[32] S.Sohail et al, "Explainable and optimally configured artifical neural networks for attack detections in smart homes", *arXiv preprint arXiv:2205.080443v1,*2202.
[33] T. Jamgharyan, "Research of the data preparation algorithm for training generative-adversarial network", Bulletin of High Technology, no. 19, pp. 40-50, 2022.
[34] Kaggle datasets base website. [Online]. Available https://www.kaggle.com/datasets
[35] Registry of Open Data on AWS website. [Online]. Available

https://registry.opendata.aws/

[36] Public data sets for testing and prototyping. [Online]. Available https://docs.microsoft.com/en-us/azure/azure-sql/public-data-sets?view=azuresql

[37] Datasets base website. [Online]. Available http://apolloscape.auto/

[38] Datasets of overhead imagery. [Online]. Available http://xviewdataset.org/#dataset

[39] Google open images dataset. [Online]. Available https://ai.googleblog.com/2016/09/introducing-open-images-dataset.html

[40] MalwareBazaar Database. [Online]. Available https://bazaar.abuse.ch/browse/

[41] Malware database. [Online]. Available http://vxvault.net/ViriList.php

[42] A free malware repository for researches. [Online]. Available https://malshare.com/

[43] Malware repository. [Online]. Available https://avcaesar.malware.lu/

[44] Malware repository. [Online]. Available https://www.virusign.com/

[45] Viruses repository. [Online]. Available https://virusshare.com/

[46] A live malware repository. [Online]. Available https://github.com/ytisf/theZoo

[47] F.Wang et al, "An efficient unsupervised domain adaptation deep learning model for unknown malware detection", *International conference on security and privacy in new computing environments (SPNCE),* vol. 423, pp. 64 -76, 2022.

[48] G. Pitolli et al, "MalFamAware: automatic family identification and malware classification through online clustering", *International Journal of information security* vol. 20, pp. 371-386, 2021.

[49] S. David, R. Anand, V. Jeyakrishnan and M Niranjanamurthy, *"Security issues and privacy concerns in industry 4.0 applications",* Wiley, Beverly, 2021.

[50] I. Priyadarshimi and R.Sharma, *"Artifical Intelligency and Cybersecurity",* CRC Press Taylor&Francis Group, New York, 2022.

[51] Encyclopedia by Kasperky. [Online].Available https://encyclopedia.kaspersky.ru/glossary/indicator-of-compromise-ioc/

[52] Nettitude labs web site. [Online].Available https://labs.nettitude.com/blog/context-triggered-piecewise-hashing-to-detect-malware-similarity/

[53] S.Kumar and Sudhakar, "*MCFT-CNN: Malware classification with-tune convolutional neural networks using traditional and transfer learning in IoT",* DOI 10.1016 Future Generation Computer systems, vol.25 pp. 334-351, 2021.

[54] C.Rong et al, "TransNet: Unseen malware variants detection using deep transfer learning", International Conference on Security and Privacy in communication systems (LNICST) vol.336, pp.84-101, 2020.

[55] R.Mortier et al, "Distributed data analysis", *arXiv preprint arXiv:*.2203.14088.2021.

[56] D.Pogorelov et al, "Comparative analysis of the Levenstein and Dameray-Levenstein edit distance algorithms", *Processing of Moscow State University after N.Bauman,* vol. 31, pp. 803-811, 2019.

[57] ssdeep software project website. [Online].Available https://ssdeep-project.github.io/ssdeep/index.html

[58] Professional information and analytical resource dedicated to machine learning, pattern recognition and data mining. [Online].Available http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_%D0%9C%D1%8D%D1%82%D1%8C%D1%8E%D1%81%D0%B0

[59] Capsule networks paperspace. [Online]. Available https://blog.paperspace.com/capsule- networks/

[60] Free service that analyzes malware. [Online].Available https://www.virustotal.com/
[61] Malware scanning platform. [Online].Available https://www.herdprotect.com/
[62] "Dotfuscator" software web pages. [Online].Available https://docs.microsoft.com/ru-ru/visualstudio/ide/dotfuscator/capabilities?view=vs-2022
[63] "Guardsquare" software web site. [Online]. Available https://www.guardsquare.com/proguard

Կապսուլային նեյրոնային ցանցով օբֆուսկացված վնասաբեր ծրագրային ապահովման հետազոտում

Թիմուր Վ. Ջամղարյան

Հայաստանի ազգային պոլիտեխնիկական համալսարան
e-mail: t.jamgharyan@yandex.ru

Ամփոփում

Ներխուժման հայտնաբերման և կանխարգելման համակարգերը ցանցային ենթակառուցվածքի անվտանգության ապահովման անբաժանելի բաղադրիչն են: «Դասական» ներխուժման հայտնաբերման և կանխարգելման համակարգերը չեն կարողանում հայտնաբերել այնպիսի սպառնալիքներ, որոնք նկարագրված չեն համակարգի կանոններում: Բացի այդ, նաև բաց խնդիր է համարվում օբֆուսկացիայի ենթարկված վնասաբեր ծրագրային ապահովման հայտնաբերումը:
Ծրագրային ապահովման և ցանցային ենթակառուցվածքի անվտանգությունով զբաղվող հետազոտողները, փորձում են նշված խնդիրը լուծել մեքենայական ուսուցման միջոցով: Հետազոտությունում ներկայացված են փոխանցման ուսուցման մեթոդով ուսուցանված կապսուլային նեյրոնային ցանցի ցուցաբերած արդյունքները վնասաբեր ծրագրային ապահովման հայտնաբերելու հարցում: Հետազոտությունը իրականացվել է վնասաբեր ծրագրային ապահովման ելակետային կոդի հիման վրա, կիրառելով համատեքստա-մասնատված հեշավորման մեթոդը: Վնասաբեր ծրագրային ապահովման ելակետային կոդերը ստացվել են հանրահասանելի աղբյուրներից: Կապսուլային նեյրոնային ցանցի ուսումնասիրության արդյունքները համեմատվել են նախապես ուսուցանված փաթույթային նեյրոնային ցանցի և վնասաբեր ծրագրային ապահովման հայտնաբերելու հանրահասանելի համացանցային ծառայությունների միջոցով: Մշակված ծրագրային ապահովման ելակետային կոդերը, նախապես ուսուցանված մոդելը, տվյալների հավաքածուների մի մասը, հոդվածում չներառված հետազոտության արդյունքները հասանելի են https://github.com/T-JN կայքում:

Բանալի բառեր՝ կապսուլային նեյրոնային ցանց, անորոշ հեշավորում, ներխուժման հայտնաբերման համակարգ, խմբագրական հեռավորություն, ցանցային ենթակառուցվածք:

Исследование обфусцированного вредоносного программного обеспечения с помощью капсульной нейронной сети

Тимур В. Джамгарян

Национальный политехнический университет Армении
e-mail: t.jamgharyan@yandex.ru

Аннотация

Системы обнаружения и предотвращения вторжений являются неотьемлимым компонентом безопасности сетевой Инфраструктуры. Классические системы обнаружения и предотвращения вторжений не в состоянии обнаружить угрозу не описанную в наборе правил. Также нерешенной полностью задачей является: задача обнаружения вредоносного программного обеспечения подвергнутого обфускации.
Исследователи в сфере безопасности программного обеспечения и сетевой Инфраструктуры пытаются решить данные задачи с помощью машинного обучения.
В работе представлены результаты исследования использования трансферного обучения капсульной нейронной сети для обнаружения вредоносного программного обеспечения. Исследование проводилось на основе исходного кода вредоносного программного обеспечения с использованием метода контекстно-кусочного хеширования. Исходные коды вредоносного программного обеспечения были получены из общедоступных источников программного обеспечения. Проверка результатов обучения капсульной нейронной сети проводилась с использованием обученной сверточной нейронной сети и общедоступных источников тестирования вредоносного программного обеспечения. Исходные коды разработанного программного обеспечения, часть наборов данных для обучения нейросети, результаты исследования не внесенные в статью представлены по адресу https://github.com/T-JN

Ключевые слова: капсульная нейронная сеть, нечеткое хэширование, система обнаружения вторжений, редакционное расстояние, трансферное обучение.