The model retrieves software behavior information using a hierarchical model of nested auto-associating neural networks

Martovytskyi Vitalii[1]

Ruban Ihor[2]

Bukin Ihor[3]

Smyrnov Lev[3]

[1]Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine, vitalii.martovytskyi@nure.ua

[2]Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine, ihor.ruban@nure.ua

[3]Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine, lev.smyrnov@nure.ua

Abstract. *The existing algorithms for searching for malicious code are imperfect. For the most part, search programs determine the presence of malicious code based on already known samples. Thus, as a part of the study, a model for extracting information from data was developed, containing hierarchically related basic elements based on deep noise-canceling auto-encoders. In addition, the requirements for the model and the necessary operation parameters were determined.*

Keywords: *auto-encoder, SAE, auto-associating neural networks, malware.*

I. Introduction and Problem statement

The existing algorithms for searching for malicious code are imperfect. For the most part, search programs determine the presence of malicious code based on already known samples.

One of the drawbacks of this approach is the need to obtain a copy of the malware before extracting the pattern necessary for its future detection. Obtaining a copy of new or unknown malware usually entails infection or attack on a computer system.

To complicate matters, an increasing number of malware variants are automatically generated per day. A recent Symantec report [1] shows that the number of new malware at the beginning of February 2019 increased by 36 percent compared to last year, and the total number of samples exceeded 500 million.

Most modern virus attacks today are aimed at specific targets. Thus, the number of corporate infections in 2019 increased by 12% compared to 2018.

Modern technologies allow the use of artificial intelligence systems and data mining in almost any field of computer science, and, in particular, in information security.

The purpose of this report is to describe a mathematical model capable of extracting information about the behavior of software using a hierarchical model of nested auto-associating neural networks, which is resistant to non-informative transformations and is able to highlight the most important features of malicious software.

II. Problem solution and Results

In recent years, neural networks have proven successful in creating invariant representations for complex data [2], and the presented method attempts to use similar principles to generate invariant representations for malicious programs.

The basis of the proposed model is represented by neural networks grouped in layers $L_0, L_1, \ldots, Lc$ and connected sequentially with each other. Each layer is a fragment of a hierarchical chain, receiving as input data the data processing results, transformed to the lower level presentation layer, and sending the output data to the next level of the model.

Each layer is trained separately, implementing level-by-level model training. After training the first level of the model, the input data for the next layer is presented in the form of transformation results of the previous level that was trained earlier (output of the presentation layer of the previous model level). All levels are trained in the same way on the data transformed by the previous levels of the model, after which the process is repeated for the next levels.

The main component of the model is an auto-encoder neural network that functions separately within one hierarchy level of the model. An auto encoder consists of three main components — an encoder, a representation layer, and a decoder.

To train the auto-encoder, all parts of the auto-encoder are used, after which, the components of the encoder and representation layer are separated from the auto-encoder. Representation layers are later called upon to generate a compressed representation for the next level of the model.

The structural diagram of the model is shown in Fig. 1.

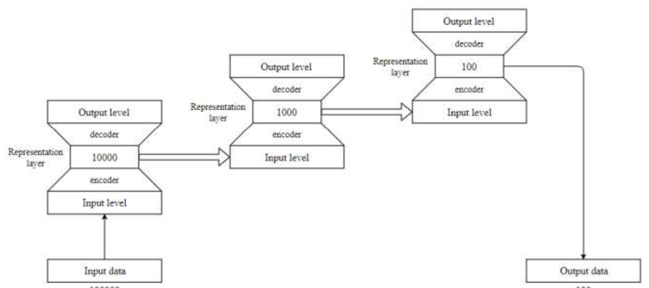

Figure 1. Structural diagram of the applied SAE architecture

Using the model to create a signature proceeds as follows. Let there be a desired recognition function $g\colon X \to Y$, the argument of which is a data set characterizing an aspect of the behavior of malware $x_n \in X$, presented as a vector of length n, and the values of functions are a set of values approximating the aspect of behavior of malware $y_m \in Y$, varying depending on the task.

There is a subset of pairs of arguments and values of the function $D = \{\{x_0,y_0\},\ldots,\{x_m,y_m\}\}$. The presented model thus implements the function $h{:}X \to Y$, which would approximate the function g over its entire domain of definition, including the values outside of subset D[3].

To calculate the value of $h\,(x)$, a data sample (vector) is fed to the input of the first layer of the trained model, and then a sequential transformation is performed while maintaining information content at each level of the model. Output activation at the last representation level of model is a vector with a lower dimension that defines a brief characteristic of the behavior of the sample based on a given aspect of the behavior.

At the same time, it is supposed to train models on various data sets characterizing various aspects of the behavior of the software instance. Thus, the aggregate characteristic of the behavior of an instance will consist of a composition of the results of all models.

Thus, the model is characterized by the following parameters:

– the number of representation levels $L_0, L_1, \ldots, L_c$;

– the required size of the last representation layer dim ($L_c^{n\,//\,2\,+\,1}$), where $dim(L)$ – the function for determining the dimension of the layer;

– the number of inner layers of each level of the model, consisting of coding L_i^e, representation L_i^r and decoding L_i^d layers, which are organized in the following order: $L_i^e{}_1$, ..., $L_i^e{}_m$, L_i^r, $L_i^d{}_1$, ..., $L_i^d{}_m$, and the total number of which will be equal $L_i n\_layers = 2m+1$;

– the function of reducing the dimension on each layer of the model $fR(Li)$, which returns the size of the representation layer required from the level;

– internal parameters of each level of the model:

- the number of neurons in each layer of each level;
- data normalization parameters;
- parameters for adding noise to the training data of the layer;
- parameters of layer activation functions;
- error function parameters;
- the number of epochs of learning level.

The number of representation levels is a parameter that is calculated dynamically, depending on the function of dimension reduction. The $fR(L_i)$ function takes the model level as an input, and calculates the size of the representation layer. It should be noted that for the first level of the model, the dimension of the first layer should be equal to the dimension of the input data, and for each subsequent layer, the dimension of the input layer will be equal to the result of calculating the function fR for the previous level:

$$\dim(L_0) = len(X)\,, \tag{1}$$

Where $len(X)$ – the dimension of the input data,

$$\dim(L_i) = fR(L_{i-1}). \tag{2}$$

As part of the work, the following dimension reduction functions were implemented:

$$fR(L_i) = \frac{\dim(L_i^0)}{c}, \tag{3}$$

where c – certain constant, the function $\dim(L)$ – the function of determining the dimension of the layer, and L_i^0 – the input layer of the level.

Also, parameters calculated at runtime include the coefficient of dimension reduction on each layer of the model level (Per-Layer Reduction or PLR), which characterizes how many times the number of coding part neurons will decrease on each new layer and, accordingly, how many times will increase the number of neurons in the decoding layer:

$$\begin{cases} c*1000, & \dim(L_i^0) > c*1000 \\ \dim(L_i^0) - c*100, & \dim(L_i^0) > c*100 \\ \dim(L_i^0) - c*10, & \dim(L_i^0) > c*10 \end{cases} \tag{4}$$

n_layers – the number of layers at a given model level. With this, the number of neurons in each layer of each level is determined.

Learning curves for model levels are shown in Fig. 2.

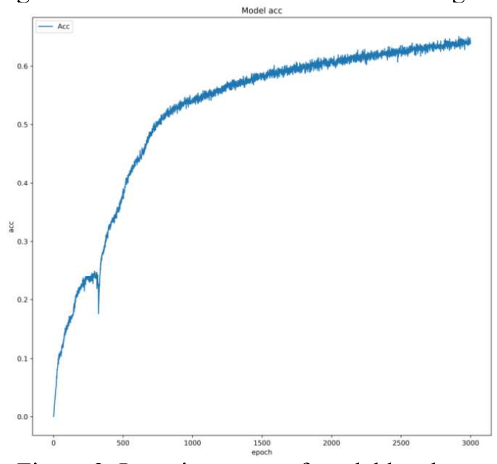

Figure 2. Learning curve of model levels

The model for processing the sample data consists of two layers of auto-encoders, which consistently reduce the dimensions 1006 > 250 > 50. Each level of this model was trained during 3000 training epochs, as a result, the average reconstruction accuracy is approximately 60%.

III. Conclusions

Thus, as a part of the study, a model for extracting information from data was developed, containing hierarchically related basic elements based on deep noise-canceling auto-encoders. In addition, the requirements for the model and the necessary operation parameters were determined.

IV. References

[1] 2019 Internet Security Threat Report, ISTR Volume 24 [Электронный ресурс] // Symantec – 2019, - access: https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-executive-summary-en.pdf – (data: 05.04.2020).

[2] Ruban, I., Martovytskyi, V., & Lukova-Chuiko, N. (2018). Approach to Classifying the State of a Network Based on Statistical Parameters for Detecting Anomalies in the Information Structure of a Computing System. *Cybernetics and Systems Analysis*, *54*(2), 302-309.

[3] Ruban, I. V., Martovytskyi, V. O., Kovalenko, A. A., & Lukova-Chuiko, N. V. (2019, September). Identification in Informative Systems on the Basis of Users' Behaviour. In 2019 IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL) pp. 574-577.