Sparse Autoencoders for Unsupervised Netflow Data Classification

Rafał Kozik(✉), Marek Pawlicki, and Michał Choraś

UTP University of Science and Technology in Bydgoszcz, Bydgoszcz, Poland
rafal.kozik@utp.edu.pl

Abstract. The ongoing growth in the complexity of malicious software has rendered the long-established solutions for cyber attack detection inadequate. Specifically, at any time novel malware emerges, the conventional security systems prove inept until the signatures are brought up to date. Moreover, the bulk of machine-learning based solutions rely on supervised training, which generally leads to an added burden for the admin to label the network traffic and to re-train the system periodically. Consequently, the major contribution of this paper is an outline of an unsupervised machine learning approach to cybersecurity, in particular, a proposal to use sparse autoencoders to detect the malicious behaviour of hosts in the network. We put forward a means of botnet detection through the analysis of data in the form of Netflows for a use case.

1 Introduction

Network and information security is presently one of the most pressing problems of the economy, a major concern for the citizens, a serious matter for the contemporary society and a crucial responsibility for homeland security. However, the immense growth of Internet users generated a plethora of adversaries who abuse the Internet's framework. Currently, the number of successful attacks on information, citizens, and even secure financial systems is still growing. Some attacks are performed by malicious users acting alone, some are carefully arranged invasions performed by groups of compromised machines. In this paper, as a use case scenario, we consider the problem of botnet detection by means of analysing the data in form of NetFlows. The problem of botnets is related to the situation where massive numbers of computers have been infected, through an array of methods, like e-mail attachments, drive-by downloads etc. The infected machines from a kind of network controlled by a botmaster, who issues a fire signal to cause malicious activities. The problem of botnets is highly relevant, as these can be responsible for DoS attacks, spam, sharing or stealing data, fraudulent clicks and many other. NetFlow, often abbreviated to simply flow is a derivative of a data stream shared between two systems. It records comprise a statistic of traffic between the same IP addresses, same source and destination ports, IP protocols and IP Types of service. The reason, why researchers invest efforts to develop the mechanism for NetFlow-based detection techniques is of two-fold. Firstly, the

M. Choraś and R. S. Choraś (Eds.): IP&C 2018, AISC 892, pp. 192–199, 2019.
https://doi.org/10.1007/978-3-030-03658-4_23

nature of the NetFlow data structure allows overcoming the problems related to the privacy and sensitiveness of information that underlying data can exhibit [3]. Secondly, NetFlow standard is widely used by the network administrators as a useful tool for a network traffic inspection purposes.

The remainder of the paper is organized as follows: in Sect. 2 the state of the art in network anomaly detection is given. Section 3 contains the description of the autoencoder-based unsupervised approach to cybersecurity, whereas in Sect. 4 the results obtained on malware datasets are presented and discussed. The paper is concluded with final remarks and plans for future work.

2 Related Work

One of the challenges of combating the botnets is to identify the botmaster in order to block the communication with the infected machines. Currently, the malware is using Domain Generation Algorithms (DGA). DGAs are a way for botnets to hide the Command and Control (C&C) botmaster server. When the botmaster server is compromised, it loses command over the entire botnet. Therefore, anti-virus companies and OS vendors blacklist its IP and stop any possible communication at the firewall level. In [2] a NetFlow/IPFIX based DGA-performing malware detector is suggested. DGA malware is expected to attempt to try to contact more domains than it does new IP addresses. Because of the NetFlows unidirectionalness, additional information has to be used to single out the originator of each transmission. NetFlows with the same IP-port-protocol triples are paired and marked as request and response according to timestamps since a request always comes first. However, this method loses its reliability when scaled to larger networks. In such cases, a service detection algorithm provides a strong feature based on a median number of peers difference. A DNS anomaly detector is implemented, labeling the right tail of the normal distribution as anomalous. This is because DNS requests that are more numerous than the visited IPs are possible C&C botnet connections. Anomaly values are acquired with a fuzzy function. Finally, as the proposed method is susceptible to raising a false positive for DNS resolver service, data from the service detection step is used to tackle this problem.

A real-time intrusion detection system (IDC) with a hardware-core of High-Frequency Field Programmable Gate Arrays was presented in [4]. The authors examine a batch of IDC's including the recent additions to the field, which are based on computational intelligence, including fuzzy systems, support vector machines, and evolutionary algorithms like differential evolution, genetic algorithms or particle swarm optimisation. Whereas these software-based methods are able to quickly adapt to new threats, their effectiveness in high-volume environments is limited by their detection speed. This translates into the inability to properly address the needs of massive environments in the near future, like cloud computing. The proposed hardware-cored IDS offers a higher detection speed than the software-based counterparts. The method supplies a Field-Programmable Gate Array (FPGA) with an internally evolvable Block Based

Neural Network (BBNN). The BBNN in this process is a feed-forward algorithm. The character of the blocks and the weights are determined by a genetic algorithm which seeks a global optimum guided by a specific fitness function. NetFlow data is used as a way to streamline feature extraction, as these can be set to default flow features. Additionally, the NetFlow collector is able to generate real-time data for the FPGA. The procedure itself is as follows: the FPGA performs real-time detection of possible intrusions and adds the record to the database; the BBNN repeatedly re-trains itself with the fresh database; the FPGA corrects its configuration building on the structure of the BBNN.

In [5] authors proposed the use of a micro-cluster based outlier detection algorithm (MCOD) as a deviation discovery device, augmented to take into account pattern deviation over time. The procedure employs clustering to cut down on the number of distance calculations it has to perform. The reduced calculation needs make it suitable for real-time data stream analysis, unlike many other anomaly detection approaches. The distance between a flow and a cluster centroid decides whether it is an anomaly, or not. MCOD is used in a succession of intervals, making the algorithm time-aware. The effects of the anomaly detection are then processed by a polynomial regression to arrive at an approximation of cluster densities over time. In the proposed procedure, all cluster densities are monitored disregarding the cluster edge denoted by the k variable. This approach diminishes the impact of the unlikely supposition that all traffic is distributed equally across the network, allowing for an increased situational awareness. By comparing cluster densities over time two polynomials are generated to represent the cluster activity. Overall, the proposed method flags anomalies in two distinctive ways. The MCOD detects distance-based divergences at the end of every time series. The polynomials created over 3-h and 24-h periods, when compared using Frechet distance, reveal any anomalies of actual versus expected behaviour of a cluster.

On the other hand, a deep learning for real-time malware detection has been analysed in [6]. Signature-based approaches form the current industry standard for malware detection, despite obvious shortcomings with detecting obfuscated malware, zero-day exploits and simply the astounding daily number of new malware releases. To tackle these kinds of problems, anomaly detectors based on machine learning models are implemented. Methods like K-nearest neighbour, support vector machines, or decision tree algorithms struggle with high false positive rates. Without sufficient context, malware classification is hard to perform accurately. On the other hand, Deep Learning (DL) algorithms are adequate for making superior decisions, but at a cost. DL needs significantly more time to retrain the detection model, which constitutes a major drawback when new malware strains have to be added frequently. The proposed procedure tries to strike a balance between the accuracy of deep learning and the swiftness of classical machine learning methods by employing a multi-stage detection algorithm cooperating with the operating system. The first stage involves classical machine learning detection. In case the ML classifies a potential threat, it is carried over to the 'uncertain stage'. In the second stage, a Deep Learning algorithm decides

if the threat is marked as benign or as hostile, and therefore, killed. If new malware is found, the model is retrained with the use of a concept-drift component, which makes sure the model is relevant.

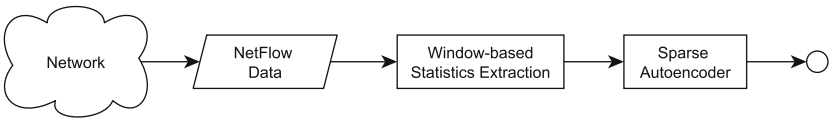

Fig. 1. The general overview of the proposed system architecture.

3 Proposed System Architecture

The general architecture of the proposed solution has been shown in Fig. 1. Conceptually, the data is collected from the network in a NetFlows format. It captures aggregated network properties. Commonly, that kind of data is collected by network elements and later sent to the collectors. The statistics retrieved from NetFlows are often used by network administrators for auditing purposes. Single NetFlow aggregates statistics (e.g. number of bytes sent and received) about packets that have been sent by specific source address to a specific destination address. Usually to capture the long-term malicious behaviour of a specific node, additional analysis of the data is required. In the proposed approach we calculate statistical properties of a group of NetFlows that have been collected for a specific source IP address within a fix-length time spans called time windows. These statistics include:

- number of NetFlows,
- number of source ports,
- number of protocols used,
- number of destination IP addresses,
- number of destination services,
- sum of bytes exchanged between source and destination,
- total number of exchanged packets.

4 Sparse Autoencoder Overview

The Autoencoder is a type of an artificial feedforward neural network that is trained in an unsupervised manner. During training, the target values are set to be equal to the input values and the backpropagation algorithm is utilised. The architecture of the network consists of three layers – one hidden layer, and two visible ones (see Fig. 2).

Typically, for classical Autoencoder, the number of hidden neurons is lower than the number of neuron in visible layers, so that the network is trained to

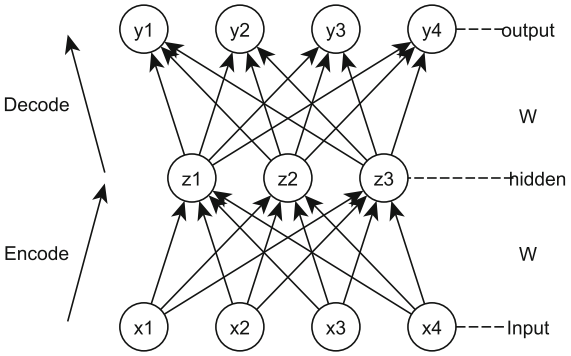

Fig. 2. The structure of the auto encoder.

learn the low-level representation of the underlying data. In case of a Sparse Autoencoder, the number of hidden neurons is significantly higher than in the visible layers. In such case, the network is trained to learn a sparse representation. Typically, bringing the data into a higher dimension reveals interesting facts about the data and allows achieving better classification results. Typically, for the Autoencoders the tied weights concepts is used. It means that the neural network weights for encoding and decoding are the same. The responses of hidden units are calculated using formula (1), where σ indicates sigmoid activation function ($\sigma(x) = \frac{1}{1+exp(-x)}$), N number of visible units, M number of hidden units, c hidden biases, b output layer biases, and w the weights of the artificial neural network.

$$z_j = \sigma\left(\sum_{i=1}^{N} w_{ij}x_i + c_j\right) \tag{1}$$

The input data is reconstructed using (2) formula. The parameters to be trained are weights w and biases vectors b and c.

$$y_i = \sigma\left(\sum_{i=1}^{M} w_{ji}z_j + b_i\right) \tag{2}$$

In order to train the Autoencoder the negative log likelihood loss function is optimized with respect to w, b and c. The loss function has following form:

$$E(x_i, y_i) = -\sum_{i=1}^{N} [x_i ln(y_i) + (1 - x_i) ln(1 - y_i)] \tag{3}$$

Calculating $\frac{\partial E}{\partial w_{ij}}$ for such defined loss function, it is easy to show that weight in k-th iteration can be updated using following formula:

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} + \gamma\left\{\left(\sum_{i=1}^{N} w_{ij}^{(k)}(x_i - y_i)\right) z_j(1 - z_j)x_i + (x_i - y_i)z_j\right\} \tag{4}$$

where γ indicates learning rate. Similarly, we calculate updates of biases, using following formulas:

$$c_j^{(k+1)} = c_j^{(k)} + \gamma \left\{ \left(\sum_{i=1}^{N} w_{ij}^{(k)} (x_i - y_i) \right) z_j (1 - z_j) \right\} \tag{5}$$

$$b_i^{(k+1)} = b_i^{(k)} + \gamma (x_i - y_i) \tag{6}$$

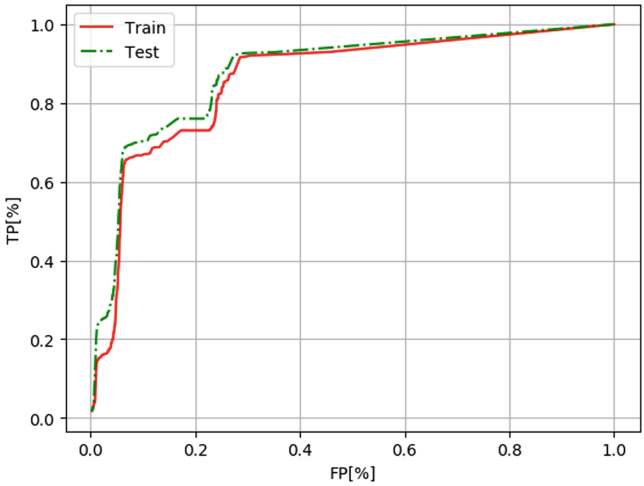

Fig. 3. Effectiveness of detected anomalies (TP) versus number of false positive alerts (FP).

5 Experiments

In order to evaluate the proposed solutions, we have used datasets provided by Malware Capture Facility Project [1]. The datasets are fully labelled and each corresponds to different scenarios of malware infections and/or botnet activities. The dataset has been divided into training and evaluation parts. In practice, the detection algorithm is trained on a subset of scenarios and tested on the others. It is a more real-life strategy than a commonly used approach where part of a specific scenario is used for training and the remaining part for testing. In order to tune the parameters of the model we additionally split the training dataset into 5 parts and performed 5-fold cross-validation.

It must be noted that we perform system training only on the normal traffic. The autoencoder is explicitly never exposed to the traffic samples which include malicious behaviour. This simplifies the learning process since the administrator does not have to label the data. The experiments showed that the proposed

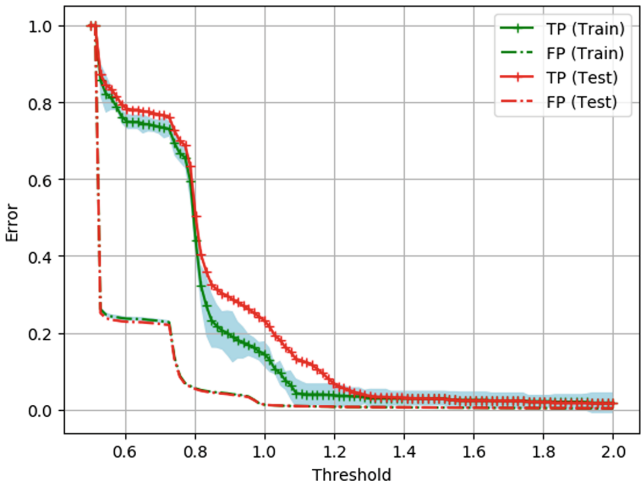

Fig. 4. Effectiveness metrics (for training and validation datasets) versus detection threshold.

solution can achieve 80% of detection effectiveness while having 20% of false positives. The results are comparable both for the data we used for training and for unknown traffic we used for validation (testing).

The average performance of the proposed method has been shown in Fig. 3. It can be noticed that both for training and testing datasets the results are comparable. Moreover, the errors (True Positive and False Positives Rates) with respect to the changing threshold have been shown in Fig. 4. The green band on the figure indicates the range of three standard deviations for the training dataset.

6 Conclusions

In this paper, we have demonstrated the proposal of the unsupervised machine learning approach to cybersecurity. The proposed solution has been used to detect the malicious behaviour of hosts in the network. We have analysed a specific demonstration use case, where the problem of botnet detection by means of analysing the data in form of NetFlows is considered. The presented results are promising.

References

1. The Malware Capture Facility Project. https://mcfp.weebly.com/
2. Grill, M., Nikolaev, I., Valeros, V., Rehak, M.: Detecting DGA malware using NetFlow. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, pp. 1304–1309 (2015). https://doi.org/10.1109/INM.2015.7140486

3. Abt, S., Baier, H.: Towards efficient and privacy-preserving network-based botnet detection using NetFlow data. In: Proceedings of the Ninth International Network Conference (INC 2012) (2012)
4. Tran, Q.A., Jiang, F., Hu, J.: A real-time NetFlow-based intrusion detection system with improved BBNN and high-frequency field programmable gate arrays. In: IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, pp. 201–208 (2012). https://doi.org/10.1109/TrustCom.2012.51
5. Flanagan, K., Fallon, E., Awad, A., Connolly, P.: Self-configuring NetFlow anomaly detection using cluster density analysis. In: 19th International Conference on Advanced Communication Technology (ICACT), Bongpyeong, pp. 421–427. https://doi.org/10.23919/ICACT.2017.7890124
6. Yuan, X.: PhD forum: deep learning-based real-time malware detection with multi-stage analysis. In: IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, pp. 1–2 (2017). https://doi.org/10.1109/SMARTCOMP.2017.7946997