2L-ZED-IDS: a Two-Level Anomaly Detector for Multiple Attack Classes

Marta Catillo, Massimiliano Rak and Umberto Villano

Abstract Cloud computing is currently a thriving technology. Due to their critical nature, it is necessary to consider all kinds of intrusions and abuses that typically plague cloud environments. In order to maintain its resilient-state, a cloud system should have tools capable of detecting known and updated threats, but also unknown attacks (0-day). This paper presents a two-level deep learning architecture for detecting multiple attack classes. In particular, it is an extension of a previous study with a dual objective: reducing the false alarm rate and improving the detection rate, and testing the system with different types of attacks. The problem is treated as a semi-supervised task, and the anomaly detector exploits deep autoencoder building blocks. The model is described and tested on the recent CICIDS2017 and CSE-CIC-IDS2018 datasets. The performance comparison with our previous study shows a lower false alarm rate and the validity of the model for multiple attack classes.

1 Introduction

Cloud Computing is an extensively used technology. According to the NIST definition, it is a model that allows to enable ubiquitous, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) rapidly and with minimal effort [14]. Due to its growing popularity, cloud computing faces a plethora of different problems including

Marta Catillo
Dept. of Engineering, University of Sannio, Benevento, Italy, e-mail: marta.catillo@unisannio.it

Massimiliano Rak
Dept. of Computer Engineering, University of Campania Luigi Vanvitelli, Aversa, Italy, e-mail: massimiliano.rak@unicampania.it

Umberto Villano
Dept. of Engineering, University of Sannio, Benevento, Italy, e-mail: villano@unisannio.it

security ones. Therefore, it is crucial to consider all the aspects regarding the security level provided by such a paradigm. Although some security issues are well known, in such a large and evolving context it is necessary to constantly reconsider the validity of traditional protection mechanisms in order to deal with increasingly sophisticated attacks. All cloud systems, due to their distributed nature, could be vulnerable to new attacks and intrusions [17]. As stated in [15], security is a challenge in cloud computing; as a matter of fact, intrusion detection tools for cloud systems are becoming increasingly refined, with the aim of managing large volumes of data and detecting unknown malicious behavior. Therefore, the study of new detection techniques with high accuracy and low false positive rate is an active research topic. Among recent works, modern machine learning techniques, and in particular deep learning, have proven to be surprisingly useful [11].

In this paper we present a deep learning anomaly detector for multiple attack classes. In particular, it is an extension of our previous study of an anomaly detector for DoS attacks that treats the problem as a semi-supervised task and uses the autoencoder model. More specifically, semi-supervised learning combines supervised and unsupervised learning, in that it requires at some extent data labeling [5]. The model presented in our previous study identifies abnormal network traffic and flags the relative data points as outliers. We achieved 95.82% accuracy for DoS attacks on the CICIDS2017 dataset and we showed the possibility to use the autoencoder model to discover DoS attacks not present in the training set, such as 0-day attacks [4]. Although these results are encouraging, there are a number of reasons that led us to extend it. In particular:

- *Reducing False Alarm Rate*
  in our previous study, the false alarm rate, equal to 4.32% for DoS attacks, was not negligible. In order to reduce this value, we introduce now a two-level technique. This is a training approach linked to the well-known concept of *double-loop learning*, typically used in a completely different domain to create and transfer knowledge within an organization [2]. Our objective is to reduce the number of false positives thanks to the two-level approach;
- *Extending to multiple attack classes*
  the model proposed in the previous work was extensively tested on DoS attacks. However, cloud services could be vulnerable also to other types of attack, and it is worth performing more extensive testing;
- *Testing with updated datasets*
  in the previous study, we used the CICIDS2017 dataset. For the sake of completeness, in this work we will also consider the results obtained with the updated version of the dataset, CSE-CIC-IDS2018.

The remainder of this paper is structured as follows. Section 2 deals with related work. The paper will go on by illustrating our basic approach and our new two-level proposal (Section 3). Section 4 presents the results obtained. Finally, the conclusions are drawn and our future work outlined.

2 Related Work

As mentioned in the introduction, the anomaly detector presented in this paper is the evolution of a previous prototype, based on semi-supervised learning and on an autoencoder model. Both the new and the original proposal can be classified as *NADS (Network Anomaly Detection Systems)* based on machine learning [16]. For space reasons, here we will not deal with the literature related to this class of anomaly detectors, referring the interested reader to the "Related work" section of our previous paper [4]. In the following, the focus will be only on intrusion detection systems for cloud environments, and on the two-level training technique that is one of the distinctive features of our new proposal.

The literature contributions on intrusion detection challenges and opportunities for clouds are surveyed in [13], where the authors describe the different types of IDS in cloud environments and discuss possible intrusion detection techniques.

In [9], the authors propose an innovative intrusion detection system (IDS) for cloud computing based on a combination of a multilayer perceptron (MLP) network, an artificial bee colony (ABC), and fuzzy clustering algorithms. They use the CloudSim simulator and the NSL-KDD dataset to test their system. Even if this IDS is machine learning-based, it is different from our proposal, which exploits simply deep autoencoders and not a combination of different components or algorithms.

Idhammad *et al.* [10] propose a distributed machine learning intrusion detection system for cloud environments. Their system is designed to be inserted in the cloud, so as to intercept incoming traffic to the edge network routers. The authors use an ensemble of Random Forest classifiers to perform a multi-class attack classification. The paper presents the results of experiments conducted on the CIDDS-001 (Coburg Intrusion Detection Data Set) dataset. This is a flow-based dataset created in a cloud environment based on OpenStack platform. It is relatively recent but, unlike the CICIDS datasets we used for our tests, does not allow to distinguish between individual attacks. DoS attacks, for example, are not subdivided in multiple classes. The authors achieve an average accuracy of 97% and an average running time of 6.23 s for all the dataset.

A study presenting a learning approach similar to our two-level training technique, but in a different application domain, is reported in [7]. The authors perform iterations of *Double Clustering (DC)*, a two-stage clustering procedure. The first DC iteration extracts a meaningful structure of the data, while a number of the successive iterations gradually improve the clustering quality. The authors achieve remarkable results on text categorization tasks, as their unsupervised procedure can be competitive with a supervised support-vector machine. The double clustering technique was first presented in [19], where the advantages of DC over other clustering methods are highlighted and commented.

3 2L-ZED-IDS: a Two-Level Anomaly Detector

3.1 Basic approach

The core component of our anomaly detector is the *autoencoder (AE)*. Autoencoders, which are a particular type of ANN, are trained to reconstruct their input vector. They are composed of an input layer, an output layer, and one or more hidden layers. The input and output layers have the same dimension while the hidden layer typically has a smaller dimension than that of the input. In particular, the hidden layer learns the latent representation of input vectors in a different feature space with smaller dimensions [8]. If multiple hidden layers are used, the resulting network is known as *deep* or *stacked autoencoder*.

The learning process forces the AE to catch most relevant features of training data at the hidden layer, also called *bottleneck*, in such a way that the input can be reconstructed at the output layer. The autoencoding process consists of encoding and decoding. During the encoding phase, the autoencoder tries to represent the given input by using its hidden layer(s). In the decoding phase it tries to reconstruct the input by using the information encoded in its hidden layer(s). The training process aims at reducing the Reconstruction Error (RE), defined as the difference between the reconstructed and the original version of the input [3].

As mentioned before, our solution is based on the use of a (stacked) autoencoder for anomaly detection. In particular, we assume that the traffic representing the attacks is an abnormal deviation from standard network traffic. We train and validate our autoencoder by learning from a training dataset containing only "normal" network traffic. This model, capable of reconstructing inputs corresponding to normal traffic, is successively used to identify any behavioral anomalies (outliers) attributable to attacks. If an AE is trained using only "normal traffic" data, it will provide a low RE (good reconstructed representation) for any normal input data, and high RE (bad reconstructed representation) for anomalous input data. This approach is represented graphically in Figure 1.

The discrimination between anomalous and normal inputs can be based on the use of a suitable threshold RE value (*anomaly threshold*): the inputs producing RE values under the threshold are considered "normal", and those with REs above the threshold anomalous ones. A key activity that drives the whole process to set up the trained autoencoder, to find the threshold, and to test its performance on a suitable dataset is the partitioning of a labeled dataset. This process is described in detail in our previous work [4]. As a final result, we get the ZED-IDS AE, a network that can be used for attack detection. If input data leads to an RE value higher than the fixed threshold, the system will launch an alert.

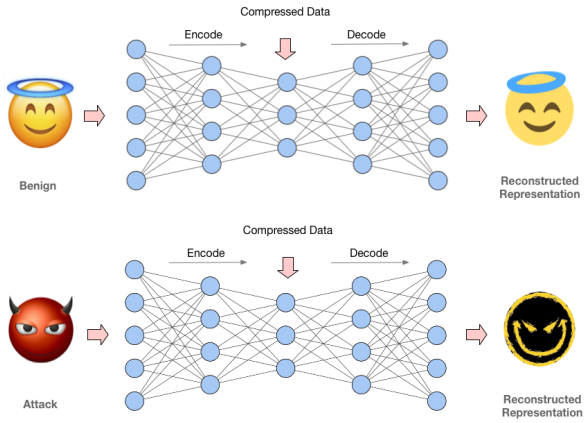

Fig. 1 Attack detection by a trained autoencoder

3.2 Two-Level approach

In the field of intrusion detection reducing the number of false positives is an active research topic, as many false alarms are typically generated during the intrusion detection process. This issue mainly concerns anomaly-based IDS, where in most cases it is very hard to establish an appropriate and accurate "normal" profile. Our previous prototype ZED-IDS is an anomaly-based detector, and produces a non-negligible number of false positives. Not surprisingly, we recorded a false alarm rate equal to 4.32% for DoS attacks [4].

As mentioned above, in the context of anomaly-based detection the problem of false positives arises because it is difficult to properly profile benign and therefore normal behaviors. Starting from these considerations, we re-evaluated our model by providing a double level of learning. As discussed in the introduction, this approach partially derives from the well-known concept of *double loop learning* [2], a technique that involves changing objectives and decisions-making rules based on experience. In particular, *double loop learning* entails the modification of goals or decision-making rules in the light of experience. In out context, the results of a trained autoencoder are used to train two successive autoencoders, in order to obtain a two-level network with improved detection capabilities. Our goal is to minimize the number of false positives and possibly also to increase the detection rate by reducing the number of false negatives. It is worth pointing out that this methodology only partially derives from *double loop learning*, as the objective remains the same throughout the whole process.

Starting from these considerations, we defined the two-level process sketched in Figure 2, where:

- *Network flow*: is single system input (flow) to be evaluated;

- *Net1, Net2Ok, Net2KO*: are ZED-IDS autoencoders obtained by different training sets (more on this later);
- *OK*: are network flows classified as BENIGN;
- *KO*: are network flows classified as ATTACK;
- *False negatives (FN)*: are ATTACK network flows classified as BENIGN;
- *False positives (FP)*: are BENIGN network flows classified as ATTACK.

Our goal is to use the upper AE at the second level to distinguish BENIGN flows (OK) from false negatives (FN), which are to be classified as BENIGN and ATTACK, respectively. On the other hand, the lower AE at the second level is used to distinguish ATTACK flows (KO) from false positives (FP), which are to be classified as ATTACK and BENIGN , respectively.

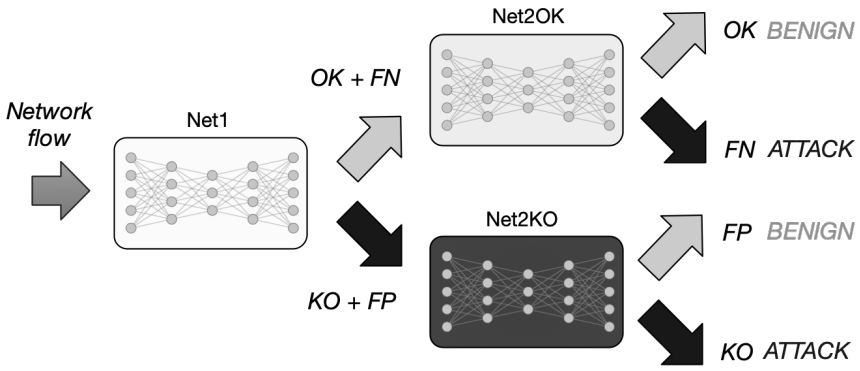

Fig. 2 2L-ZED-IDS: operating model

In particular, *Net1* is trained with network flows contained in the training dataset, according to the single-level approach defined in the above section. *Net2OK* and *Net2KO*, instead, are trained with output data from the (previously trained) *Net1* autoencoder, *OK + FN* and *KO + FP*, respectively. It is worth pointing out that the three autoencoders (*Net1*, *Net2OK* and *Net2KO*) share the same structure of the ZED-IDS AE. The only difference between them lies in the data used for their training.

We have tested the two-level networks, with the subset of 20% flows from the original dataset not used in the multiple training/validation phases previous carried out. The results presented in the next section show that our two-level approach has a significant impact on the overall performance of the system, as reduces the false alarm rate and slightly improves the detection rate as compared to the single autoencoder model tested in our previous work.

4 Results

For the evaluation of our approach, we chose as a reference the modern CICIDS2017 dataset [18]. In fact, we carried out tests on both the 2017 dataset and the most recent version of 2018 (CSE-CIC-IDS2018). These datasets are publicly available and contain both benign traffic and many known recent attacks. The data are available in packet format (PCAP) and flow labeled format (CSV). In particular, for the latter format, each record is a *labeled flow* resulting from the network traffic analysis carried out by the tool CICFlowMeter [12]. Each record is a flow identified by 85 features. These include network traffic characteristics as well as labeling indicating whether the flow is a benign or an attack one.

The main difference between the two versions of the dataset concerns the environment in which they were created. The 2017 version was created in a simulated context with a few machines, while the 2018 attacking infrastructure version includes 50 machines and the victim organization has 5 departments with 420 machines and 30 servers. The CSE-CIC-IDS 2018 dataset was collected on the Amazon AWS computing platform, and so it is also known as CIC-AWS-2018 dataset. There are also some differences in sample sizes between the two versions of the datasets. In the CSE-CIC-IDS 2018 dataset there is a larger number of samples for most attacks. The attacks in the two versions of the dataset are the same. However, some of them have been made by exploiting different tools. These differences are sketched in the Figure 3 that shows all the attacks performed in the two versions of the dataset.

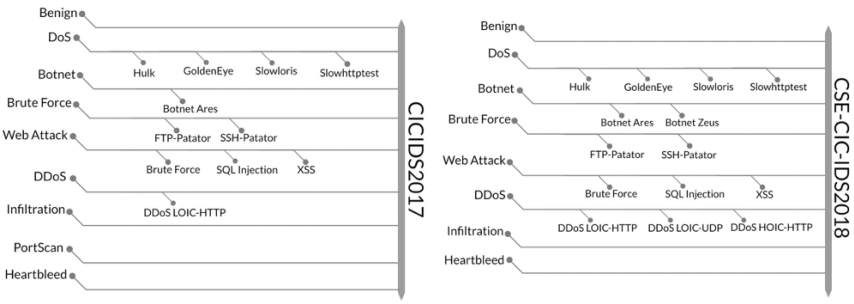

Fig. 3 CICIDS2017 and CSE-CIC-IDS 2018 datasets

During the first phase of experimentation we performed data preprocessing. As mentioned above, each flow is identified by 85 features, but we only use 83 attributes for our analysis, since the `Flow_ID` and `Timestamp` feature are not relevant for the detection process. We implemented the 2L-ZED-IDS with Python, Keras [6] and Tensorflow [1]. Customarily, the performance metrics are computed by means of the obtained total number of true positives (TP), of true negatives (TN), of false positives (FP) and of false negatives (FN). From these four values, it is possible to compute the following metrics, widely used in the intrusion detection field:

- *Detection Rate*: ratio of the number of correctly classified anomalous instances to the number of all actual anomalous instances:

$$Detection\ Rate = \frac{TP}{(TP+FN)} \tag{1}$$

- *Accuracy*: ratio of the number of correctly classified anomalous and normal instances to the number of all instances:

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \tag{2}$$

- *Precision*: ratio of the number of correctly classified anomalous instances to the total number of instances that are classified as anomalous:

$$Precision = \frac{TP}{(TP+FP)} \tag{3}$$

- *False Alarm Rate*: ratio of the number of incorrectly classified normal instances to the total number of all actual normal instances:

$$False\ Alarm\ rate = \frac{FP}{(FP+TN)} \tag{4}$$

Table 1 shows the results obtained using the single-level ZED-IDS approach. It is worth pointing out that for some types of attacks, such as *Infiltration*, there is no measurement. This is because there are only 36 *Infiltration* samples in CICIDS2017 dataset, while *PortScan* attacks are not present in CSE-CIC-IDS2018 dataset. *Heartbleed* attacks, on the other hand, were not considered as they are an unrepresentative number of samples in the original dataset (11 samples in CICIDS2017). Table 2 instead shows the results obtained by applying the two-level approach. A comparative analysis between the two approaches is reported in Table 3. In particular, it is possible to note that there is a slight improvement in the detection rate after the application of the two-level approach, but what is most interesting is the significant reduction of the false alarm rate which tends to drop by more than 50% in almost all cases. This shows that the double learning approach can lead to an overall improvement in performance, especially regarding false positives. It is also important to point out that our AE has been trained in about 300s, and the detection time for a single flow is about 1 microsecond.

5 Conclusions

Anomaly detection is an active research topic in the security and cloud systems industry. In this context, it is essential to find countermeasures capable of mitigating the possible damage to the infrastructure caused by an attack. In this paper, we have presented the 2L-ZED-IDS anomaly detector, as an extension of our previous work.

Table 1 Performance Comparison - Single-Level Approach

Attack	Detection rate %		Accuracy %		Precision %		False alarm rate %	
	CIC-2017	*CIC-2018*	*CIC-2017*	*CIC-2018*	*CIC-2017*	*CIC-2018*	*CIC-2017*	*CIC-2018*
DoS	95.8	98.2	95.7	96.2	92.7	91.9	4.3	4.9
Botnet	96.9	97.3	98.6	98.5	90.1	90.3	1.1	1.3
Brute Force	97.6	98.2	91.2	91.6	90.2	90.1	2.2	2.1
Web Attack	98.5	93.7	98	96.7	90	90.6	2	2.5
DDoS	96.9	97.5	96.7	97.3	97.6	98	3.4	2.8
Infiltration	-	96.5	-	96.8	-	98.3	-	2.7
PortScan	96.2	-	95.9	-	96.5	-	4.4	-

Table 2 Performance Comparison - Two-Level Approach

Attack	Detection rate %		Accuracy %		Precision %		False alarm rate %	
	CIC-2017	*CIC-2018*	*CIC-2017*	*CIC-2018*	*CIC-2017*	*CIC-2018*	*CIC-2017*	*CIC-2018*
DoS	98.1	98.8	98	98.7	98.7	97.8	1.1	1.3
Botnet	98.6	98.9	99.3	99.2	94.8	95	0.5	0.6
Brute Force	98.2	99.2	95.2	95.8	94.9	94.8	1.1	1
Web Attack	99	97.8	98.9	98.6	94.7	95.7	1	1.1
DDoS	97.7	98.7	97.9	98.7	98.8	99	1.7	1.2
Infiltration	-	98.4	-	98.5	-	99.1	-	1.3
PortScan	98.1	-	97.9	-	98.2	-	1.2	-

Table 3 Performance Comparison - Single-Level Approach/Two-Level Approach

Attack	Detection rate % 1-L		Detection rate% 2-L		False alarm rate% 1-L		False alarm rate % 2-L	
	CIC-2017	*CIC-2018*	*CIC-2017*	*CIC-2018*	*CIC-2017*	*CIC-2018*	*CIC-2017*	*CIC-2018*
DoS	95.8	98.2	98.1	98.8	4.3	4.9	1.1	1.3
Botnet	96.9	97.3	98.6	98.9	1.1	1.3	0.5	0.6
Brute Force	97.6	98.2	98.2	99.2	2.2	2.1	1.1	1
Web Attack	98.5	93.7	99	97.8	2	2.5	1	1.1
DDoS	96.9	97.5	97.7	98.7	3.4	2.8	1.7	1.2
Infiltration	-	96.5	-	98.4	-	2.7	-	1.3
PortScan	96.2	-	98.1	-	4.4	-	1.2	-

It is an anomaly detector machine learning-based and characterized by an innovative learning approach in the context of intrusion detection, *double learning*. The results obtained show the potential of the technique. We achieved a maximum value for the detection rate of 99.2%, but the most significant improvement is the reduction of the false alarm rate which, in the best case, reaches 0.5%. This reduces the effect of the false alarm rate of our previous proposal.

It is important to point out that all tests have been carried out on a laptop without GPU acceleration. Training and testing times can be further reduced using dedicated and/or parallel hardware GPU-equipped. In our future work, we plan to analyze flows relative to non-synthetic traffic collected on real networks in cloud environments. We also plan to apply "fine-tuning" on the learning steps of the entire system (possibly increasing the number of levels) with the aim of further reducing the number of false positives.

References

1. Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015). URL http://tensorflow.org/
2. Argyris, C.: Double loop learning in organizations. Harvard Business Review **55**(5), 115–125 (1977)
3. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
4. Catillo, M., Rak, M., Villano, U.: Discovery of DoS attacks by the ZED-IDS anomaly detector. Journal of High Speed Networks **25**, 349–365 (2019)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3) (2009)
6. Chollet, F., et al.: Keras. https://github.com/fchollet/keras (2015)
7. El-Yaniv, R., Souroujon, O.: Iterative double clustering for unsupervised and semi-supervised learning. In: T.G. Dietterich, S. Becker, Z. Ghahramani (eds.) Advances in Neural Information Processing Systems 14, pp. 1025–1032. MIT Press (2002)
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning (2016)
9. Hajimirzaei, B., Navimipour, N.: Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. ICT Express **5**(1), 56 – 59 (2018)
10. Idhammad, M., Karim, A., Belouch, M.: Distributed intrusion detection system for cloud environments based on data mining techniques. Procedia Computer Science **127**, 35 – 41 (2018)
11. Keegan, N., Ji, S.Y., Chaudhary, A., Concolato, C., Yu, B., Jeong, D.H.: A survey of cloud-based network intrusion detection analysis. Human-centric Computing and Information Sciences **6**, 1–16 (2016)
12. Lashkari, A.H., Zang, Y., Owhuo, G., Mamun, M.S.I., Gil, G.D.: Cicflowmeter (formerly iscxflowmeter) — a network traffic flow analyzer. URL http://www.netflowmeter.ca/netflowmeter.html
13. Mehmood, Y., Shibli, M.A., Habiba, U., Masood, R.: Intrusion detection system in cloud computing: Challenges and opportunities. In: 2013 2nd National Conference on Information Assurance (NCIA), pp. 114–125 (2013)
14. Mell, P., Grance, T.: The NIST definition of cloud computing. NIST Special Publication **800**, 145 (2011)
15. Moctar, C.B.O.M.E., Konat, K.: A survey of security challenges in cloud computing. In: 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 843–849 (2017)
16. Moustafa, N., Hu, J., Slay, J.: A holistic review of network anomaly detection systems: A comprehensive survey. Journal of Network and Computer Applications **128**, 33–55 (2018)
17. Riaz, A., Qadir, J., Younis, U., ur Rasool, R., Ahmad, H.F., Kiani, A.K.: Intrusion detection systems in cloud computing: A contemporary review of techniques and solutions. J. Inf. Sci. Eng. **33**, 611–634 (2017)
18. Sharafaldin., I., Lashkari., A.H., Ghorbani., A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proc. of the 4th Int. Conf. on Information Systems Security and Privacy - Volume 1: ICISSP,, pp. 108–116. INSTICC, SciTePress (2018)
19. Slonim, N., Tishby, N.: Document clustering using word clusters via the information bottleneck method. In: Proc. of the 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p. 208215. Association for Computing Machinery, New York, NY, USA (2000)