Improving Apache Spot Using Autoencoders for Network Anomaly Detection

A. Priovolos, G. Gardikis, D. Lioprasitis, S. Costicoglou
R&D Department
Space Hellas S.A.
Athens, Greece
{apriovolos;ggar}@space.gr

***Abstract*—Apache Spot is an increasingly popular opensource platform for advanced network insights, focusing on the detection and analysis of anomalies, which can potentially correspond to security incidents. In this paper, we propose an improvement over Apache Spot's built-in Machine Learning algorithm (Latent Dirichlet Allocation - LDA), replacing it with an Autoencoder based on deep learning techniques. We implement the Autoencoder functional block and deploy it into the Apache Spot's pipeline, integrating it with Hadoop and Spark. Finally, we evaluate and benchmark the Autoencoder against the built-in LDA, using a publicly available network traffic dataset with cyber-attacks. The result is a considerable increase of accuracy, precision and recall.**

Keywords: Autoencoder, Apache Spot, Cybersecurity, Security Analytics, Latent Dirichlet Allocation

I. INTRODUCTION

Over the past years, the frequency of cyber-attacks in networks has increased. These attacks have become more complex, causing more damage to their victims. Anomaly-based detection systems attempt to detect deviations from what is considered as the normal behavior. Anomaly detection is embedded as a feature in most modern security analytics platforms. Open-source platforms also exist, such as Apache Spot [1] and Apache Metron [2]; in this paper, we will focus on the former. Apache Spot is a platform for advanced network insights, focused on detecting and analyzing cyberthreats. As an alternative to LDA, in this paper we propose the use of an Autoencoder for detecting anomalies in network traffic. We integrate our Autoencoder in the Apache Spot system for operational use, replacing the existing LDA algorithm. Finally, we benchmark our proposed model comparing it with Spot's built-in LDA algorithm, using IDS 2018 [3] dataset, which consists of real traffic and attacks inside a local network.

II. BACKGROUND, MOTIVATION AND RELATED WORK

Apache Spot [1] is an end-to-end anomaly-based detection system. It consists of three main components: (i) Ingest component, which collects data from multiple sources and saves them in the deployed Hadoop Distributed File System (HDFS), (ii) Machine Learning (ML) component, which uses LDA [4] on top of Apache Spark to score connections and (iii) Operational Analytics (OA) component, which visualizes through its web UI the network's topology and the suspicious connections.

Machine learning and deep learning techniques are widely used, among others, for anomaly detection problems [5-10]. Taking this work into account, and towards improving performance, we propose the implementation, integration into the Spot pipeline and subsequent evaluation of an Autoencoder. Autoencoders compress the given input into a smaller representation and then they try to decompress it and reproduce the input. The error between the given input and its output is very small under normal circumstances. If the error is above a threshold the connection is considered as abnormal and marked as suspicious.

III. IMPLEMENTATION

Our Autoencoder has been deployed in the Machine Learning component of Apache Spot, replacing Spot's built-in LDA. Each given input to the algorithm represents a network flow. The input contains 18 metrics (protocol and flags, ports, duration, number and size of incoming and outgoing packets) from flow records, as they are described in section III.B.

The architecture of the Autoencoder is simple. It consists of three layers: The first layer is the input layer, composed by 18 neurons. The second layer is composed by 12 neurons. This layer encodes the input and creates a compressed representation. As activation function for this layer we have selected to use tanh. We selected to use tanh, instead of ReLU which is often used, because our data are zero-centered with range [-1, 1]. Tanh has range (-1, 1), but ReLU on the other hand has range [0, Infinity). The third layer is the output layer and is composed by 18 neurons. In this layer, the data are decoded and it tries to rebuild the input of the model. For training our Autoencoder, we have selected to use Adam optimizer, and train it for 90 epochs. We have built our Autoencoder on top of Spark using Elephas [11], a Keras extension for deploying deep learning models distributed using Spark. With this library we can take advantage of Spark's parallel processing to run our Autoencoder without the need of bespoke hardware and GPUs.

IV. EVALUATION

A. Dataset

Spot is fully capable of working with live feeds from operational networks. However, for the sake of evaluation, it is very difficult to create a live feed which corresponds to realistic network usage and also includes cyber attacks. For this

TABLE I. AUTOENCODER & LDA RESULTS

Day in Dataset	Autoencoder				LDA			
	Acc (%)	*Precision (%)*	*Recall (%)*	*F1*	*Acc (%)*	*Precision (%)*	*Recall (%)*	*F1*
14/02	95.5	11.9	100.0	0.21	51.4	3.04	28.3	0.05
15/02	95.4	8.3	87.4	0.15	52.4	0.8	46.8	0.01
16/02	95.5	11.9	100.0	0.21	51.6	0.82	13.8	0.01
20/02	94.8	34.2	77.0	0.47	50.4	1.7	16.8	0.03

purpose, we used a captured dataset which we manually fed into the ingest component of Spot. In specific, we have used the IDS 2018 [3] dataset. This contains network traffic based on HTTP, HTTPS, SMTP, POP3, IMAP, SSH and FTP protocols, captured for 10 days. All days are containing attack records. The captured attacks can be divided in five categories: bruteforce attacks, DoS and DDoS attacks, web attacks, infiltration attack and botnet attack.

Autoencoders must be trained with normal traffic only in order to be able to detect attack traffic as abnormal. Because there is not a single day that contains only benign traffic, we have selected a day and we have filtered only the benign traffic in order to create our training set. We have used 20% of the training set for validation. For testing our Autoencoder, we have used the rest days of the dataset.

Due to the ratio of benign and malicious flows in the network traffic of the day we have selected to train our model, we have chosen the 95th percentile in order to identify outliers. We test our Autoencoder with the benign traffic of the day we select for training and we calculate the threshold for outliers as follows: from these records, we calculate the 95th percentile of their scores. We consider as malicious record every record that its score is bigger that the calculated threshold. This same threshold has also been used in LDA algorithm in order to identify outliers in network traffic.

B. Results

For testing our Autoencoder and comparing it with Spot's LDA algorithm we have used IDS 2018 dataset. We have observed that for bruteforce, DoS and DDoS attacks our Autoencoder performs very well and outperforms Spot's LDA. In Table 1 we show the performance metrics of these two algorithms for these types of attacks for each of the days of the dataset. The improvement in accuracy, but even more in recall and precision, is obvious. In Fig. 1, we show the F1 score of our Autoencoder compared with Spot's LDA algorithm.

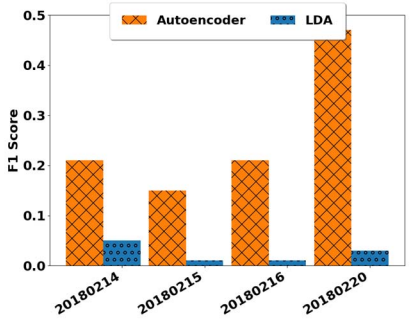

Fig. 1 F1 Score of Autoencoder & LDA

For the remaining days of the dataset the other types of attacks are based on a botnet scenario, *i.e.* Botnet attack, or in an known exploitation in application level, *i.e.* Web attack, SQL injection, or even in human factor, *i.e.* Infiltration attack where the user is assumed to download a malicious executable. For these types of attacks, the performance of both Autoencoder and Spot's LDA algorithm is significantly worse. Even if they both exhibit good accuracy and very high recall (51% - 100% for Autoencoder and 38%-50% for LDA) the precision they can achieve os very low (around 0.1% for Autoencoder and almost 0% for LDA). This happens because of the extremely low ratio of malicious vs. benign flow records in the dataset. This results in a considerably high number of false positives.

V. CONCLUSION AND FUTURE WORK

This paper describes our proposal for improving Apache Spot's performance. We implemented an Autoencoder, replacing the existing LDA algorithm. We have evaluated our proposal using a real-life dataset, which contains network traffic and some attack records. We have tested our model with this dataset and we have shown that Autoencoder outperforms by far Spot's built-in LDA in all cases.

As future steps, we plan to train our model to learn from sequences of records, instead of one record at a time. In addition, we plan to create a classification algorithm for suspicious connections to assist the human operator to better recognize and counter the threat. In the list of future features we have also included the ingest of various metrics from the network infrastructure, beyond plain network flow data, e.g. metrics from core and edge cloud and also the access network, to detect and counter novel threats in complex infrastructures, including 5G networks.

Last but not least, towards increasing performance for large-scale deployments, we plan to exploit the technical developments of the EU-funded EVOLVE project [12]. EVOLVE is building a high-performance testbed by combining Big Data, Cloud and HPC technologies. We have already included our developments as one of the Proof-of-Concept use cases to be implemented in the project. In specific, we will deploy the Autoencoder-based solution in the EVOLVE testbed in order to take advantage of the hardware and software features offered for boosting performance and improving scalability.

ACKNOWLEDGMENTS

The work described in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 825061 (EVOLVE) and No 815178 (5GENESIS).

REFERENCES

[1] Apache Spot, https://spot.apache.org/
[2] Apache Metron, https://metron.apache.org/
[3] IDS 2018 Dataset, University of New Brunswick, https://www.unb.ca/cic/datasets/ids-2018.html
[4] D. Blei, A. Ng, M. Jordan, "Latent dirichlet allocation", Journal of machine Learning research, vol. 3, pp, 993-1022, 2003.
[5] R. Chalapathy, A. Menon, S. Chawla, "Anomaly detection using oneclass neural networks", arXiv preprint arXiv:1802.06360, 2018.
[6] D. Hendrycks, M. Mazeika, T. Dietterich, "Deep anomaly detection with outlier exposure", arXiv preprint arXiv:1812.04606, 2018.
[7] O. Aydin, S. Tasabat, "Outlier Detection Method by Using Deep Neural Networks", PressAcademia Procedia, vol. 5, number 1, pp. 96-101.
[8] R. Chalapathy, S. Chawla, "Deep learning for anomaly detection: A survey", arXiv preprint arXiv:1901.03407, 2019.
[9] M. Du, F. Li, G. Zheng, V.Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning", SIGSAC, 2017.
[10] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. Siddiqui, A. Binder, E. Muller, M. Kloft, "Deep one-class classification", Deep oneclass classification, pp. 4393-4402, 2018.
[11] Elephas, http://maxpumperla.com/elephas/.
[12] EVOLVE project, HPC and Cloud-enhanced Testbed for Extracting Value from Diverse Data at Large Scale, https://www.evolve-h2020.eu