Using Neural Network for Security Attacks Detection

1. Introduction

Software Systems and especially their data is often target of cyber criminals. Being able to detect potential malicious user behaviour on time can significantly increase the security and safety of systems and their sensitive data. However, this process is often quite complicated taking into account that there are several different types of cyber security attacks that affect a plethora of network and software system parameters that renders existing intrusion detection systems vulnerable especially when the users slightly change their attack behaviour.

In this work we have used state of the art machine learning techniques for detecting potential security attacks through the in-depth analysis of network traffic and software systems behaviour. In particular we have developed a system that internally uses neural networks (NN) for detecting potential security attacks based on the data collected so far. For this purpose we have used the NSL-KDD dataset [5] (i.e., an "improved" version of the KDDCup 99 dataset that is widely used across the research community) that contains both training and test data about several different types of security attack and hence it is rather useful for both training our models as well as their evaluation.

The document is structured as follows. Initially in Section 2 related work in this field is presented. The description of the dataset that will be used in this work follows in Section 3. The developed neural network is described in Section 4, while the evaluation of the system along with a brief discussion is presented in Section 5. Finally in the last section we summarize the main points of this work.

2. Related Work

Intrusion Detection (ID) is an important component of every security system and in parallel with firewall and antivirus software systems intend to discover, determine and identify unauthorized system or data usage, duplication or modification of data or system behaviour and overall potential system or data destruction, coming either from internal or external attacks [1]. The methods used by an ID system can be classified in two broad categories, that is, signature-based and anomaly-based ID techniques. The signature-based techniques utilize the signature (e.g., pattern) of previously identified attacks and can adequately cover system/data misusage. The anomaly-based techniques intend to find a potential attack through the analysis of the current status of systems and their deviation from their normal behaviour. They can possibly identify previously unknown attacks (e.g., zero-day attacks). However, they often produce a considerable number of false alarms.

The Machine Learning (ML) and Data Mining (DM) techniques are particularly useful for intrusion detection purposes. These techniques are often used in the context of a Knowledge Discovery System that applies different algorithms that belong to the ML or DM field for extracting data patterns that could signify a security attack. The unsupervised ML/DM techniques intend to find patterns, structure or generally knowledge in unlabeled data whereas the supervised ML/DM techniques intent to develop a function or model that understands the data and hence can decide about the presence of a security attack or not. Ideally the data collected (i.e., both labeled data from past cases and the ones that depict the current status of the system) should contain adequate information about the network status and the (operating) system behaviour. That is why the KDD Cup 1999 dataset [2] is often used in the training process (several publications highlight the usage of this dataset) despite the fact that it has been developed before more than 20 years. The most difficult and time consuming process of model development is the training process that often require a considerable amount of time. Once the model developed it can be accordingly used for detecting potential attacks either in offline or online systems.

Traditional Machine Learning (ML) techniques such as Bayesian Networks, Logistic Regression, Decision Trees and Support Vector Machines are based on a predefined list of features. They are parameters of particular interest for the design of the underline model (supervised ML techniques) or the clustering of the data in broader categories (unsupervised ML techniques). Consequently the performance of the AI systems developed is highly connected with the accuracy of the features extracted and used. [3]. Deep Neural Networks (aka Deep Learning) are multiple level representation methods directly extracted from raw data using general purpose learning procedures [4]. They can efficiently deal with high-dimensional data and discover complex patterns or hidden correlations among them. Hence they provide a valuable tool for detecting and preventing systems and resources from previously unknown or complicated security attacks such as zero-day attack and advanced persistent threats.

3. Dataset and Features

The NSL-KDD dataset [5] that will be used in this work has several advantages over the initial KDD Cup 1999 dataset. First of all, there is no redundant record in the training dataset and hence the classifiers that would be developed based on this dataset would not produce biased results. Also there is no duplicate record in the testing dataset and hence the performance of the classifiers would be unbiased. Last but not least, the number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set.

For each record there is information about 41 different features (with either discrete or continuous values) including (a) basic features (obtained from TCP/IP connections), (b) content features (extracted from data content) and (c) traffic features as well. A complete list of features is available at the following table (Table 1).

#	Feature	#	Feature	#	Feature	#	Feature
1	duration	11	num_failed_logins	21	is_host_login	31	srv_diff_host_rate
2	protocol_type*	12	logged_in	22	is_guest_login	32	dst_host_count
3	service*	13	num_compromised	23	Count	33	dst_host_srv_count
4	flag*	14	root_shell	24	srv_count	33	dst_host_same_srv_rate
5	src_bytes	15	su_attempted	25	serror_rate	35	dst_host_diff_srv_rate
6	dst_bytes	16	num_root	26	srv_serror_rate	36	dst_host_same_src_port_rate
7	land	17	num_file_creations	27	rerror_rate	37	dst_host_srv_diff_host_rate
8	wrong_fragment	18	num_shells	28	srv_rerror_rate	38	dst_host_serror_rate
9	urgent	19	num_access_files	29	same_srv_rate	39	dst_host_srv_serror_rate
10	hot	20	num_outbound_cmds	30	diff_srv_rate	40	dst_host_rerror_rate
						41	dst_host_srv_rerror_rate

Table 1: The 41 features of NSL-KDD dataset [6]. The value of features with an asterisk (*) comes from a controlled set of terms (being different for each one of them)

In both training and test datasets there is a considerable amount of records regarding the normal system behaviour. In the training dataset we can also find records from 22 different attacks that can be further organized to four main classes, i.e., Denial of Service (DoS), Probing (Probe), Remote to User (R2L) and User to Root (U2R). Nevertheless, most of them denote a DoS attack. In the test dataset there are 37 different attacks (i.e., 17 new attacks that are not mentioned in the training dataset) which render the system evaluation based on this dataset rather challenging. In Table 2 we have summarized the attacks mentioned in the train and test datasets along with the broader category they belong to.

Attack Class	Attacks
DoS	back, land, neptune, pod, smurf, teardrop, mailbomb*, apache2*, processtable*, udpstorm*
Probe	ipsweep, nmap, portsweep, satan, mscan*, saint*
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster, sendmail*, named*, snmpgetattack*, snmpguess*, xlock*, xsnoop*, worm*
U2R	buffer_overflow, loadmodule, perl, rootkit, httptunnel*, ps*, sqlattack*, xterm*

Table 2: Attacks that belong to each one of the four main categories. Attacks with an asterisk (*) are only used in the test dataset

4. Methods

4.1. Data Preparation

Both training and test data are available in CSV format and hence we can easily read them. However we should further process their data before actually use them for training our model. Initially we have separated the parameters recorded for each record from their label (i.e., the column that indicates normal system behaviour or the particular attack detected). Then we have used one-hot encoding for dealing with those parameters that their values come from a controlled set of terms (i.e., protocol type, service, flag and label). Finally we have further processed the data so that their mean is zero and scaling to unit variance.

Following the aforementioned processed two arrays of data (i.e., X and Y) were developed for both training and test datasets. The array X has 122 columns whereas the array Y has 24 columns. In the second case, there is one column for the normal system behaviour, 22 columns for each one of known system attacks (i.e., the ones mentioned in the training datasets) and a last column for any other attack. The last column is actually used to deal with unknown attacks that we can probably met in a real case scenario such as the 17 new attacks (i.e., not mentioned in the training dataset) used in the test dataset.

4.2. Model Development and Evaluation

Then we have created an Artificial Neural Network that consists of 122 input units, 500 hidden units and 24 output units (Figure 1). For hidden units the sigmoid activation function was used whereas for output units the softmax function. Accordingly we have trained the model using Regularized Mini-Batch gradient ascent algorithm with learning rate 0.5 using batches of 1000 records and we have repeated this process 10 epochs. All the aforementioned parameters were specified after performing several tests.

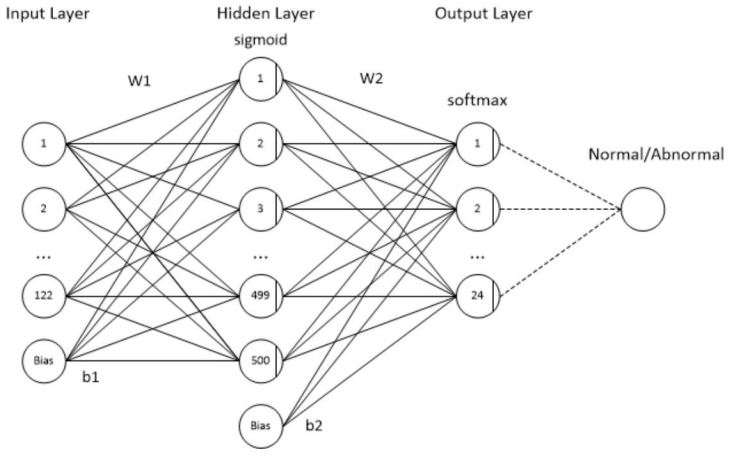

Figure 1: Artificial Neural Network Architecture for Security Attack Detection

Regarding the evaluation of the developed neural network we have computed accuracy (i.e., the total number of records successfully classified divided by the total number of records) for both training and test data. More precisely we have used the neural network developed for predicting the attack based on the given dataset and accordingly we have compared it with the expected. Taking into account that the ultimate goal of our system is to automatically detect normal or abnormal system behaviour based on the given data rather than the particular attack detected (which is also important at a latter stage) we have additionally computed the accuracy of the system on this basis. More precisely we have computed the number of times the system successfully predicted normal or abnormal system behaviour based on the total number or records.

5. Results and Discussion

5.1. Datasets and Features

In the training dataset there are many records about both normal and abnormal system behaviour. Nevertheless in the second case there are much more Dos and Probe attacks in comparison with the R2L and U2L attacks. On the other hand, in the test dataset there is not so much difference in the amount of records that belong to each class (Table 3). For instance, the amount of records about R2L attacks is greater than the records for Probe attacks.

Training Dataset		**Test Dataset**	
Attack Type	**Number of Records**	**Attack Type**	**Number of Records**
Normal	67343	Normal	9710
DoS	45927	DoS	7458
Probe	11656	R2L	2754
R2L	995	Probe	2421
U2R	52	U2R	200

Table 3: Number of Records for each class in both Training and Test datasets

In the given dataset there are 41 parameters recorded for each incident, and the value of three of them comes from a controlled set of terms. However, after using one-hot encoding we have noticed that there are 122 parameters for each record. This is perfectly normal since there are many different values (at least for parameter “service”) but it’s also obvious that this process has triplicated the amount of parameters recorded.

5.2. Neural Network Accuracy

When the training process completed we have noticed that the model developed can accurately (almost 100%) predict the appropriate attack for the records of training dataset. However, the accuracy of the model is much lower in the test dataset. In particular we have noticed that the accuracy of the model is about 70% which is to some extent normal taking into account that the dataset contains many unknown attacks. However, since we are interested to know whether the data recorded highly the presence of an attack or not we have noticed that the accuracy of the system is being increase to 80%.

The accuracy of the neural network is being improved through the time for training data as presented in the following figure (Figure 2-a). Regarding the test data the accuracy is it initially improved but at some point (after approximately 10 epochs) the accuracy of the network starts to both decrease and increase as it is also presented in the following figure (Figure 2-b). In this figure the accuracy for both training and test dataset is presented (based on the output of the Neural Network rather than the output of the whole system that indicates normal or abnormal user behaviour) using 50 epochs. Taking into account the aforementioned issue, we have specified a limited amount of epochs (i.e., 10 in our work) during the training process.

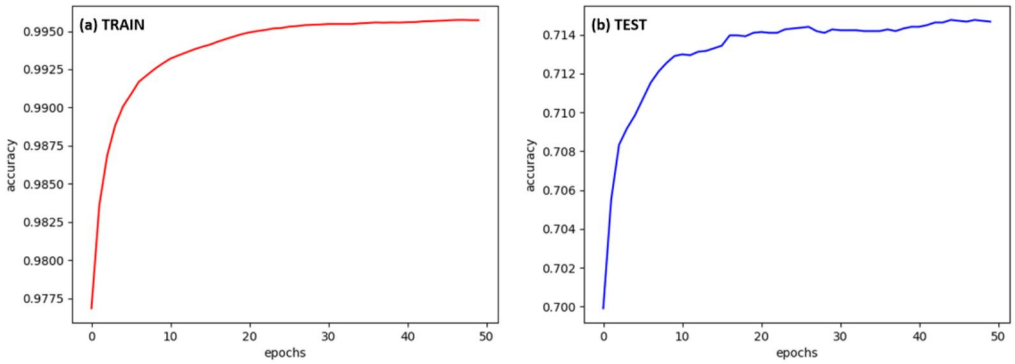

Figure 2: Neural Network Accuracy vs Epoch in (a) Training Dataset and (b) Test Dataset.

Having executed the training process several times (even with exactly the same parameters) we have also noticed that the output produced is in some cases quite different. More precisely we have noticed that in some cases, the system could predict the correct output in the test dataset with accuracy above 85%. This indicates that the model developed depends on the initialization of the parameters of the network (which in our work are random numbers) and hence inappropriate initial values may make the training process to stacks to local minimum.

6. Conclusion

Being able to detect potential malicious user behaviour is very important. However this process is still difficult since the existing systems should also deal with previously unknown attacks. This process can be facilitated through the use of machine learning techniques and especially artificial neural networks such as the one presented in this work. However such systems often produce many false alarms and hence further investigation of the incidents automatically detected by the system is necessary in order to find whether the data collected so far correspond to a true system security attack or not. In our work we have analyzed the data available at the NSL-KDD dataset and we have developed a model that it can predict potential attacks based on the data collected by the system. The model developed indicated that it can adequately predict potential attacks based on the data recorded with accuracy up to 85 percent for even new unknown security attacks. The accuracy of the system on test dataset can be further improved through the use of data coming from other datasets published so far.

7. References

[1] Buczak, Anna L., and Erhan Guven. "A survey of data mining and machine learning methods for cyber security intrusion detection." IEEE Communications surveys & tutorials 18.2 (2015): 1153-1176.

[2] KDD Cup 1999 Data, available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[3] Li, Jian-hua. "Cyber security meets artificial intelligence: a survey." Frontiers of Information Technology & Electronic Engineering 19.12 (2018): 1462-1474.

[4] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444.

[5] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.

[6] Harb, Hany M., et al. "Selecting optimal subset of features for intrusion detection systems." (2011).