Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 644–653

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Shallow and Deep Learning Approaches for Network Intrusion Alert Prediction

Mohammad Samar Ansari[a,*], Vaclav Bartos[b], Brian Lee[a]

[a]*Athlone Insitute of Technology, Athlone, Ireland*
[b]*CESNET, Czech Republic*

Abstract

The ever-increasing frequency and intensity of intrusion attacks on computer networks worldwide has necessitated intense research efforts towards the design of attack detection and prediction mechanisms. While there are a variety of intrusion *detection* solutions available, the *prediction* of network intrusion events is still under active investigation. Over the past, statistical methods have dominated the design of attack prediction methods. However more recently, both shallow and deep learning techniques have shown promise for such data intensive regression tasks. This paper first explores the use of shallow learning techniques for predicting intrusions in computer networks by estimating the probability that a malicious source will repeat an attack in a given future time interval. The approach also highlights the limits to which shallow learning may be applied for such predictive tasks. The work then goes on to show that deep learning approaches are much more suited for network alert prediction tasks. A recurrent neural network based approach is shown to be more suited for alert prediction tasks. Both approaches are evaluated on the same dataset, comprising of millions of alerts taken from the alert sharing system *Warden* operated by CESNET.

Keywords: Alert Prediction; Convolutional LSTM; Cybersecurity; Deep Learning; Gradient Boosted Decision Trees; Shallow Learning.

1. Introduction

Cyber-attacks on networks and data are becoming increasingly sophisticated and frequent. Most approaches to deal with cyber attacks can be put into the category of classifiers, since they tend to detect attacks based on classification of network packets into benign or anomalous. Some attack detection systems also go on to classify intrusion alerts based on the severity of the damaged intended. However, all such approaches are essentially *attack response*, and are reactionary in nature. Systems capable of predicting an imminent attack are still under active development. The present work explores machine learning based approaches for predicting future behavior of malicious sources.

* Corresponding author. Tel.: +353-833863655;
E-mail address: mansari@ait.ie

1877-0509
10.1016/j.procs.2020.04.070

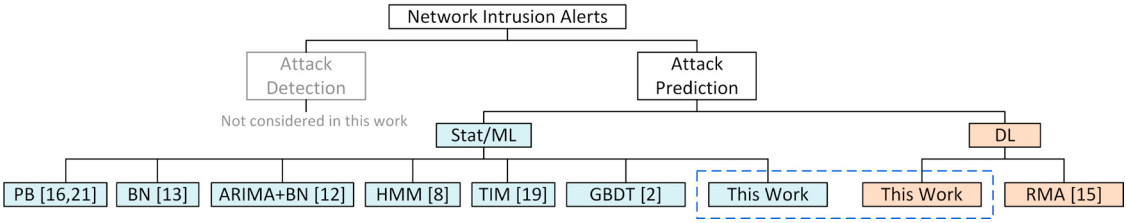

Fig. 1. Some pertinent existing works on ML-based alert prediction

Conventional machine learning (ML) approaches for IDS design belong to the category of shallow learning (SL) since these rely on manual feature engineering to craft features for the ML model [17]. Lately, there has been some research work on deep learning (DL) based IDS design. This mitigates the need for feature engineering as the deep network is expected to extract better representations of high-level data which is made possible by the inherent complex architecture of the network as well as the possibility of inclusion of non-linear transformations. There have been only a limited number of attempts to apply ML for predicting future behavior of a node. This paper first presents a SL solution for estimating the likelihood of future attacks from known malicious sources, and thereafter puts forward a DL approach for predicting future attacks by a given source, potentially against different targets.

For the SL approach, Gradient Boosted Decision Trees (GBDT) were found to be the most appropriate for the problem. Stochastic GBDT is among the most widely used ML algorithms presently. It is easy to interpret, very adaptable, and generates highly accurate models. In addition to the GBDT-based solution, fully-connected Neural Networks (NN) were also explored for the same task with very similar, albeit slightly inferior, results. For the DL based solution, choice of the network was guided by the nature of the alert data. Since the alerts coming from malicious sources are in the form of temporal sequences, it was deemed prudent to use recurrent neural networks (RNN), with their proven sequence learning properties, for the task. Long Short Term Memory (LSTM) [6] is a type of RNN which can effectively learn long term dependencies in sequence data. A special variant of LSTM known as the ConvLSTM (Convolutional LSTM) is used in the proposed DL solution for alert prediction. A ConvLSTM is similar to a conventional LSTM layer, but the input transformations and recurrent transformations are convolutional, i.e. convolutional structures are present in both the input-to-state and state-to-state transitions. A ConvLSTM with a larger transitional kernel is able to capture faster variations in feature values while one with a smaller kernel can capture slower varying trends in the data.

1.1. Motivation

The motivation of this work stems from the knowledge that while there are a wide variety of attack *detection* systems available now, there have been relatively few attempts at alert *prediction*. Moreover, the latest available work on the prediction of future security events deals with the attacks targeted towards a particular node [15], whereas the present work focuses on the prediction of alerts from a malicious source.

1.2. Contribution

The main contribution of this work is the exploration of the extents to which shallow learning and deep learning can be leveraged for alert prediction on the same dataset. First, a SL network is presented which has the capability of predicting the probability of getting an attack alert from a malicious source. Included in the discussion is the elaborate process of feature engineering for the SL method. A DL based approach is then presented with the aim of highlighting the clearly better prediction of actual alerts (as opposed to the alert probabilities in the SL case), and that too without the need for any feature engineering (and only some data pre-processing).

It is to be noted that the two approaches are actually complementary in the sense that the SL approach may first be used to identify the most probable attackers on the monitored network, and thereafter a network administrator may employ the DL method to predict future alerts originating from that attacking source.

2. Related Work

While the tasks of detection and mitigation of network intrusions received considerable attention [7], there have been relatively fewer attempts to leverage alert data for prediction of future behavior of malicious sources. One early work in that direction is a series of papers on *predictive blacklisting*, a technique in which data from a large alert sharing platform (*DShield*) is used to generate 'personalized' blacklists for each contributor, listing those sources that are the most likely to attack it in a near future [21, 16]. The works however only focus on building a blacklist, not to characterize the sources in any other way, like assigning a score or estimating the probability of different types of future alerts. More recent efforts towards prediction of malicious network activities include [13, 12, 8, 19]. Ramaki *et. al.* proposed a framework which correlates the alerts, and constructs attack scenarios using Bayesian networks (BN) [13]. Okutan *et. al.* presented a system called Cyber Attack Prediction of Threats from Unconventional Resources (CAPTURE), which used Auto-Regressive Integrated Moving Average (ARIMA) model to generate inferences from alert data, and then train a Bayesian classifier for prediction [12]. Holgado *et. al.* proposed a method based on the Hidden Markov Model (HMM) to predict multi-step attacks, and validated their method using a virtual DDoS scenario [8]. Wang *et. al.* proposed a method of cyber-attack prediction based on Threat Intelligence Modeling (TIM) wherein a matching method is used to extract threat intelligence from alert data [19]. Recently, Bartos et. al. presented a method to predict probability of observing a future attack from a given source using NN and GBDT [2]. A deep learning based solution for predicting future security events was recently proposed in [15], wherein a Recurrent Memory Array architecture was used to learn the security event streams from different machines, and output predictions about likely future attacks. The above discussion is summarized in Fig. 1.

3. Intrusion alert data

The alert data used in this work is obtained from the alert sharing system *Warden*[1], which is a platform for automated sharing of alerts amongst CSIRT teams about detected threats. A typical Warden alert contains various fields like attack category, source and target IP, port, protocol, time of attack start and end, time of detection, information about the detector and many others. For this work, the alerts were trimmed to keep only the following fields:

- `DetectTime`: time of generation of alert [numeric value]
- `Category`: attack category (e.g. `Recon.Scanning`, `Attempt.Exploit`, etc.) [categorical value]
- `FlowCount`: number of connections created or attempted by the attacker [numeric]
- `SourceIP`: /24-prefix of the attacking IP address [numeric]
- `TargetIP`: /24-prefix of the attacked IP address [numeric]
- `Protocol`: protocol used for carrying out the attack [categorical]
- `Port`: port accessed on target [numeric value, converted to categorical value (explained below)]
- `Node`: name of the detector issuing the alert [categorical]
- `Node Type`: type of the detector which issued the alert (e.g. honeypot, IDS, flow monitor, etc.) [categorical]

It is to be noted that we group source and target IP addresses with the same /24 prefix together. For the Source IPs this is motivated by the fact that hosts in the same subnetwork often exhibit similar behavior, as shown for example in the study by Moura et. al. [11] about the so-called *internet bad neighborhoods*. For Target IPs we expect it is often more likely for an attacker to target a complete subnet (e.g. an organisaton) instead of a particular IP address.

Moreover, although the `Port` field in the alerts contains numeric values, it was considered to treat it as a categorical variable, since a mismatch in the actual and predicted values of port number should be treated as an incorrectly predicted alert. The most frequently occurring 15 port numbers in the dataset were treated as independent categories, and all the other less frequent port numbers were categorized as the 'Other' category.

The dataset used for training and evaluation in this paper comprises of alerts shared via Warden during May–July 2018. It contains approximately 140 million alerts, with 6.2 million unique IP addresses (1.2 million /24 prefixes).

[1] https://warden.cesnet.cz/

4. Shallow learning approach

The first presented prediction method is based on SL using NN and GBDT with manually engineered features. The goal in this case is to estimate the probability that a given /24 prefix will perform an attack within a future time window (e.g. next day), based on information from alerts that reported the source as malicious within a history time window (e.g. last week). This kind of prediction was introduced in [2], where the probability estimation is called the *Future Misbehavior Probability (FMP) score*. This score was designed to summarize all known information about malicious IP addresses into a number. It can then be used to compare the IP addresses, sort them, create top-n lists or it can serve as one of the inputs in multi-criteria alert prioritization algorithms. In this paper, we look at it only as an alert prediction algorithm, which predicts presence of an alert (of a specific type) in the specified time window.

The method presented below is based on that from [2], but as explained above, here we only predict behavior of whole /24 prefixes, not individual IP addresses. Also, we only use information available directly from alerts (i.e. not information about presence on blacklists or geolocation, which is used in [2] to slightly improve prediction accuracy).

4.1. Prediction method

Given the distribution of alert categories in the dataset, we decided to create a separate prediction model for each of two broad groups of alerts – scanning activity (alerts with `Recon.Scanning` category) and attempts of unauthorized access (`Attempt.Login` or `Attempt.Exploit` category). Other attack types, e.g. DDoS attacks, are too infrequent in the dataset (less than 1 %) so there are not enough samples to build an accurate prediction model using this method.

In this method, an alert is represented by tuple $a = (t, e, c, v, d)$, where items correspond to the following fields: `DetectTime`, `SourceIP` (*entity*), `Category`, `FlowCount` (*volume*) and `Node Name` (*detector*). The set of all available alerts is denoted by A. The time at which prediction is computed (*current time* or *prediction time*) is marked as t_0. The *prediction window*, T_p, is the time window of length w_p immediately following t_0, $T_p = (t_0, t_0 + w_p)$. The predictor uses information about the past alerts from *history window*, $T_h = (t_0 - w_h, t_0)$, where w_h is the history window length. In the experiments below, we set $w_h = 7$ days, $w_p = 1$ day.

Since the machine learning models we are going to use are not able to process a sequence of historical alerts directly, it has to be transformed into a fixed number of manually engineered features first. So, for a given prediction time t_0, a feature vector $\mathbf{x}_{e,t_0} = (x_1, x_2, ..., x_k)_{e,t_0}$ is first computed for each source (entity) e from the alerts reporting it as malicious. The particular set of input features used in this work is:

1., 2. Number of alerts in the previous day and in the preceding week
3., 4. Total number of connection attempts (attack volume) in the past day and in the previous week
5., 6. Number of detectors reporting the address in the previous day and in the preceding week
7. EWMA[2] of number of alerts per day over the previous week
8. EWMA of total number of connection attempts per day over the last week
9. EWMA of a binary signal expressing presence of an alert (0 or 1) in each day over the last week
10. Time from the last alert
11., 12. Mean and median intervals between alerts in last week

All these features are computed separately for alerts of *scan* and *access* categories. All of them are used regardless the type of alert to be predicted, since there may be significant correlations between attacks of different categories and therefore alerts of one category may help to predict alerts of the other one. This makes 24 input features in total.

The output (class label) to be predicted, y_{e,t_0}, is binary. It depicts whether or not there is an alert reporting the source within the prediction time window:

$$y_{e,t_0} = \left\{ \begin{array}{ll} 1 & \text{if } \exists\, a \in A : a = (t, e, c, \cdot, \cdot), t \in T_p \\ 0 & \text{otherwise} \end{array} \right\} \text{where } c \text{ is the alert category (\textit{scan}/\textit{access}) to be predicted.} \quad (1)$$

Samples with $y_{e,t_0} = 1$ are deemed to belong to *positive class*, others form the *negative class*.

[2] Exponentially weighted moving average with $\alpha = 0.25$

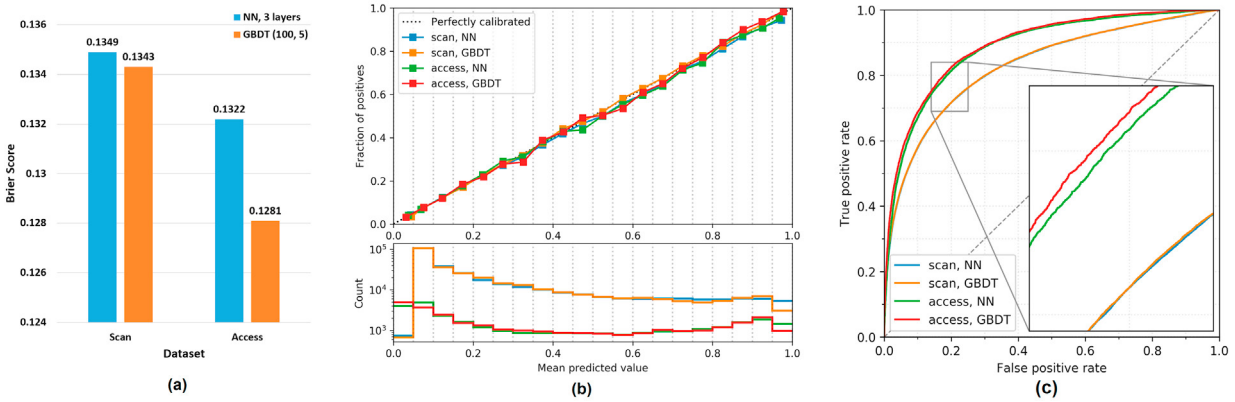

Fig. 2. (a) Brier score over testing set of *scan* and *access* datasets (b) Probability calibration curves of the models over test data. (c) ROC curves of the models over test datasets.

The ML task is to create an estimator which, for a given feature vector $\mathbf{x}_{e,t_0}$, is able to estimate the probability that $y_{e,t_0} = 1$, i.e. that the attacking source will be reported as malicious in the prediction window. This is known as *binary class probability estimation*. Output of the estimator is denoted as $\hat{y}_{e,t_0}$ and denotes the estimated probability of the positive class for the feature vector,

$$\hat{y}_{e,t_0} \approx p(y_{e,t_0} = 1|\mathbf{x}_{e,t_0}). \tag{2}$$

The metric we use to evaluate the model is Brier score (BS). For the binary case, with classes labeled 0 and 1, the BS can be described as a mean squared difference of the predicted probability of positive class and value of the real class:

$$BS = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{3}$$

where N is the number of samples. The BS can have values between 0 and 1, with lower values being better.

4.2. Dataset preparation

We evaluated the prediction method on the dataset presented in Section 3. To obtain samples for training and evaluation of the SL models we selected 18 different prediction times (t_0) within the three months. At each time t_0, we use only the IP prefixes that are reported by at least one alert (of the predicted type) within a history window of one week before t_0. For each such source, a feature vector $\mathbf{x}_i$ is compiled and a class label y_i is assigned.

This way we got 6,249,727 samples related to *scan* alerts (the *scan dataset*) and 318,940 samples related to *access* alerts (the *access dataset*). For testing, a random subset of samples is extracted from each dataset (300,000 in *scan* dataset, 30,000 in *access* dataset). The remaining data is used for training.

4.3. Results

We performed experiments with different ML models, including NN with up to 3 hidden layers and various settings of GBDT. Fig. 2(a) shows BS for the best model of each class for both datasets. The NN has 3 fully connected hidden layers, each with 24 nodes (number of input features) and rectified linear unit (ReLU) as activation function. A single node with sigmoid activation forms the output layer. The GBDT model consists of 100 trees, each with maximum depth of 5. It can be seen that all Brier scores are quite small, implying good precision of probability estimation.

The main requirement on the model is that the output actually approximates the probability of receiving another alert from the same source within the prediction window. Although this characteristic is already denoted by the Brier score, it is also possible to have a visual illustration. Figure 2(b) show probability calibration curves of the models over both datasets. These 'reliability curves' show the distribution of real classes within bins of samples with similar

estimated probability of a particular class (the positive one in our case), $\hat{y}_i$. In other words, samples are binned by the value of $\hat{y}_i$ and for each bin a point is drawn. Its horizontal position is given by the mean of $\hat{y}_i$ within that bin, vertical position equals the fraction of samples within the bin for which the true class is positive ($y_i = 1$). For an estimator which works well, i.e. its output approximates the probability of positive class, this fraction should be close to the mean of that bin, and thus the resulting line should be close to the diagonal ($y = x$). We can see that in all cases the results are very good as all the lines are indeed close to the diagonal. Histograms below the calibration curves show the number of samples in each bin.

In many use cases the estimated probability will be used along with a threshold (either fixed, or obtained by a top-n method) to split sources into 'good' and 'bad' ones, e.g. to create a blacklist. This reduces the binary class probability estimation problem into simple binary classification. In such a case, the Receiver Operating Characteristic (ROC) curves can be used to visualize results. For the evaluated models, these are presented in Fig. 2(c). An ROC curve shows the trade-off between false positive (FP) and true positive (TP) rates as the threshold value changes. For this work, TP are the sources classified as 'bad' (i.e. blacklisted) which are subsequently indeed reported as virulent within the prediction window, FP are those which are not. The closer the ROC gets to the top-left corner, the better is the classification of the model. The results of the two models on each dataset are very similar. In case of *scan* dataset they are almost identical, on *access* dataset, GBDT model is slightly better. These results agree with the Brier scores in Table 2(a). The only significant difference is between the datasets, where alerts of *access* category tend to be more easily predictable than *scan* alerts.

5. Deep learning approach

As can be observed in the previous section, the use of shallow learning requires significant amount of feature engineering. To alleviate this need for data engineering, this section proposes a viable alternative in the form of Deep Learning, which by virtue of its inherent capability to extract meaningful representations from high-level data, can be leveraged to the task of building an attack profile for an attacker, and thereafter predict future actions emanating from that source. It shall be shown in this section that the proposed DL model is able to consume the raw alert data with minimal pre-processing, and there shall be no requirement of manual feature selection and engineering.

5.1. Data pre-processing

The pre-processing steps to prepare the data for use in the DL method are shown in Fig. 3(a). First, as described in Section 3, alerts were shortened to keep only the fields corresponding to the chosen alert tuple, and Source IPs and Target IPs were trimmed to their /24 prefixes. There are therefore 13 features corresponding to one feature each for `DetectTime, FlowCount, Protocol, Node Type, Port, Category, Node`, and three features each for `SourceIP` and `TargetIP` obtained after separating the three octets in the respective /24 prefixes. Thereafter, alerts for all SourceIPs having 100 or more alerts in a month were retained and other alerts were discarded. For alerts where TargetIP is not present, a dummy 0.0.0 /24 address was inserted. It needs to be noted that for the DL method, the top-15 categories of alerts and one 'Other' category were considered as opposed to only two categories (Scan, Access) in the SL method. All the categorical variables were then converted to their equivalent Label Encoded values. For alerts with missing categorical values, a new dummy category 'Other' was created. Thereafter, a MinMax scaling was performed for each column of the alert dataset. This ensures that all the training data fed to the deep learning network is normalized between 0 and 1, and prevents large values from adversely affecting the training process. Lastly, formatted training arrays need to be generated for the processed data obtained from the previous step. Toward that end, first suitable values of the following parameters need to be chosen:

- `History`: Number of past alerts to be included in one training sample, e.g. 10 past alerts[3].
- `Future`: Number of future alerts to be included in one training sample, e.g. 2 future alerts.

[3] It may be noted that the notion of 'history' in this case is different from the one used in the previous section (SL). While here the `History` refers to a certain number of past alerts, in the previous section, history window was associated with a time value for which the alerts were considered.

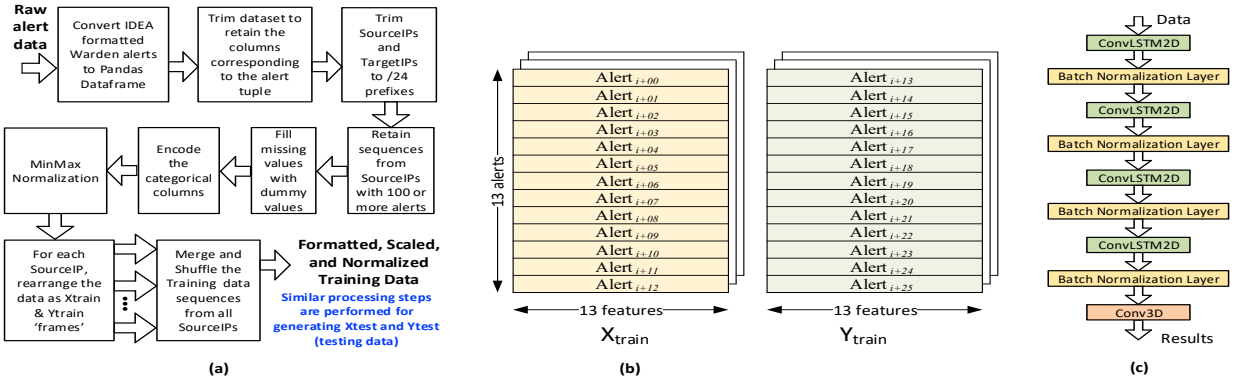

Fig. 3. (a) Data pre-processing steps. (b) Illustration of the training data for the deep learning network, *i*-th training sample. (c) Details of model

To make the data amenable for ConvLSTM, it was found prudent to format the training samples as a 2D array. Depending upon the chosen values of `History` and `Future` values, the data is rearranged in the form of a 2D array which looks very similar to a video frame. An illustration of the data formatting for training and testing purposes is provided in Fig. 3(b) from where it can be seen that both the X_{train} and the Y_{train} samples are 13×13 each. This means that during the training phase, the network is presented with a 13×13 'frame' and trained to output another 13×13 2D array. A rolling window approach is used for generating the formatted training arrays for all the individual SourceIPs, which are then merged and shuffled. Although Fig. 3(a) does not show a separate *validation* dataset at the output of the pre-processing step, it is indeed a part of the training dataset, and is separated at the time of training.

5.2. Proposed Solution

Recurrent Neural Network (RNN) architectures have been demonstrated to be very suitable for the task of learning the characteristics and patterns in time-series and sequence data [3]. Although theoretically feasible, conventional RNNs struggle when there are long-term dependencies in the data. Long Short Term Memory (LSTM) [6] was proposed as alternative to the standard RNN and is shown to be capable of addressing the vanishing gradient problem in RNNs resulting in improved long-term dependency learning. LSTMs have been applied to a variety of sequence classification and predictions tasks like Speech Recognition [5], Time Series Prediction [14], Time Series Anomaly Detection [10], Semantic Parsing [9], Human Activity Recognition [1], etc. However, not much research work has been done towards the utilization of LSTM learning characteristics for prediction of network intrusion alerts. This proposal aspires to fill that gap using a specialized variant of the LSTM viz. ConvLSTM.

The proposed deep neural network for alert prediction is shown in Fig. 3(c), from where it can be seen that the raw alert data is first pre-processed and then sent to the DL model comprising of 4 ConvLSTM2D layers with Batch Normalization layer after every recurrent layer, followed by a Conv3D layer. The training dataset is used for the training of the network and the validation set is used to estimate prediction error for model selection. After the training process is complete, a final trained model is generated which can be used to output predictions on the test dataset.

The results of the predictions obtained using the model are compared with the ground truth values in the test dataset, and the prediction performance is estimated by calculating the accuracy of the predictions as explained below.

5.3. Results

The performance of the proposed Convolutional LSTM deep neural network was evaluated on alert sequences extracted from the Warden alerts. As mentioned before, only the attacking SourceIPs which were reported more than 100 times in a month (May–July 2018) were selected for the exercise resulting in 183,959 alert sequences from different attacking sources with total alert count of 132,426,618. The training set comprised of the alerts for the months of May–June 2018, and the test set consisted of alerts for the month of July 2018. From the training dataset, 20% of the alerts were reserved for validation.

Table 1. Details of the Convolutional LSTM based deep learning model.

(a) Configuration of the different layers

Layer	Filters	Kernel Size
ConvLSTM2D Layer-1	64	(6,6)
ConvLSTM2D Layer-2	32	(4,4)
ConvLSTM2D Layer-3	28	(4,4)
ConvLSTM2D Layer-4	14	(4,4)
Conv3D Layer	1	(4,4,4)

(b) Parameters

Parameter	Value	Parameter	Value
History	13	Optimizer	Adam
Future	13	Loss parameter	MAE
Batch Size	50	Padding	Same
Epochs	50	Validation Split	0.20

5.3.1. Estimation of Prediction Accuracy

The quality of predictions is estimated by comparing predicted alerts for all the test vectors with the ground truth alerts by following the steps outlined below.

1. First, the *categorical* fields *viz.* `Node`, `Category`, `Port`, `Type`, and `Protocol` are directly compared for the predicted and actual alerts, and if there is a mismatch in *any* of these, the predicted alert is considered to be *incorrect*, without proceeding to steps 2 and 3 below.
2. For predicted alerts satisfying step-1 above, the next step is to estimate the closeness between the numeric fields in the predicted and actual alerts. For all the numeric fields a predicted alert was taken to be 'sufficiently close' to the actual alert wherever the numeric fields between the actual and predicted alert were within ±10%. However, the numeric fields for the 3 octets of the SourceIP were not provided this relaxation, as the SourceIP is trivially know to the model and should be readily reproduced in the predictions.
3. Lastly, a predicted alert is considered to be correctly predicted when all the categorical fields in the predicted alert match with their respective counterparts in the corresponding ground truth alert, *and* the values of the numeric fields are within the acceptable range defined above.

5.3.2. Prediction Performance

The configuration used for training of the model of Fig. 3(c) is provided in Table 1(a). This configuration was selected because it resulted in the best accuracy values for the predictions over the validation dataset. The various parameters set for the purpose of training are reported in Table 1(b).

The trained model was subsequently used to output predictions over the test dataset (Warden data for July 2018) which was held back during the training and validation process. This ensured that there was no leakage of information from the training phase to the testing phase. Similar to the training phase, alert data for testing was provided to the model as a 'frame' of 13 past alerts, and the model predicted the next 'frame' comprising 13 alerts. The predicted alerts were then compared with the ground truth values of the future alerts from the test dataset, and accuracy of prediction was estimated using the criteria mentioned above. Predictions were made for approximately 14 million alerts, and the average accuracy of predictions was estimated to be 63%.

Fig. 4 illustrates samples of good and bad predictions, where for the prediction labeled as 'correct', the categorical fields match with their ground truth counterparts, and the predicted numeric fields are within close proximity (±10%). For the 'incorrect' label, even some categorical fields do not match with the ground truth values (shown highlighted).

6. Discussion

6.1. Complementarity of the two methods

The previous sections presented two distinct methods for dealing with different aspects of the alert prediction task. While the SL method (Section 4) predicts the probability of observing a future alert (of a specific type) caused by a given source, the DL method (Section 5) goes further and is able to predict expected parameters of several next alerts.

For best results, the two methods can be combined in practice. For example, the SL method can be used to predict how likely it is for a source to perform an attack in a near future, and if the probability is high and there are enough previous alerts, the DL method can then be used to predict the expected parameters of such an attack. Detailed exploration of such a combined approach is left for a future research endeavor.

	Correct Prediction		Incorrect Prediction	
	Actual Alert	Predicted Alert	Actual Alert	Predicted Alert
DetectTime	2018-06-12T12:25:51Z	2018-06-12T12:03:43Z	2018-05-30T11:19:00Z	2018-05-30T11:51:21Z
FlowCount	1	1	1	821
Protocol	TCP	TCP	UDP	UDP
Type	Honeypot	Honeypot	Packet	Packet
Port	22	22	81	3389
Category	Information.UnauthorizedAccess	Information.UnauthorizedAccess	Recon.Scanning.Attempt.Exploit	Other
Node	cz.cesnet.hugo.haas_cowrie	cz.cesnet.hugo.haas_cowrie	cz.cesnet.nemea.vportscan	cz.cesnet.hugo.haas_cowrie
SourceIP-1	5	5	185	183
SourceIP-2	188	188	36	35
SourceIP-3	87	87	55	57
TargetIP-1	192	191	78	86
TargetIP-2	0	1	128	113
TargetIP-3	0	3	255	254

Fig. 4. Illustration of the actual and predicted alerts for different cases. On the left is a correct prediction and on the right is an incorrect prediction. Highlighted fields show poor/unacceptable prediction for the corresponding fields.

6.2. Comparisons with Similar Works

The SL method is based on an approach introduced in [2]. The method presented here is different in several ways, e.g. by the use of IP prefixes instead of individual addresses and the different set of input features, although the basic principles are the same. The idea of predicting probability of future attacks from a specific source was inspired by previous works on *predictive blacklisting* [21, 16].

The rest of this section focuses on comparisons between the proposed DL approach and related existing works. Early research efforts targeted towards prediction of security events mostly culminated in the assignment of probability scores to the next possible security events or alerts, and then identifying the most probable attack scenario [13, 12, 8, 19]. The DL method presented in this work however predicts the entire alert, and therefore cannot be compared directly with the methods in [4, 13, 12, 8, 19].

The closest work to this paper is Tiresias [15], which is able to predict from amongst (upto) 4495 possible events. However, Tiresias only predicts the *category* of the next event, which is a *classification* task since it predicts the most probable class out of 4495 classes (event categories). In effect, while Tiresias is predicting future events, it is indeed only able to predict the 'class' of a future attack. The proposed DL-based solution, on the other hand, predicts various parameters of the next alert, and since many of them are continuous (such as DetectTime, Flowcount, etc.), it is a *regression* task. To the best of the author's knowledge, there is no prior work which predicts exact parameters of the future alerts as done in this work, and therefore a comparison with other works cannot be drawn. Furthermore, as discussed before, being a regression task for the numeric parameters, the accuracy cannot be expected to compete with the classification based approaches.

Another underlying difference between the working of Tiresias (as well as most other alert prediction works) and the proposed method is that while existing solutions predict next attack steps against a selected target, our method predicts next attacks from a given source. While the former is suitable for hardening targets against future attacks, the latter is more suitable for building an attack profile for malicious sources and is intended to be used as part of threat intelligence platforms, reputation databases, etc.

6.3. Limitations and Future Work

The proposed methods are not without their limitations. Firstly, both the SL and the DL methods use the Warden data only. It would be interesting to test the performance of the proposed networks over data from other alert collection systems. Also, for the DL method, the choice of ConvLSTM was inspired by its ability to learn long sequences. However, there have been some recent advances in applications of 1-D Convolutional Neural Networks [20] and Attention networks [18] for the task of sequence learning. These may eventually turn out to be better suited for the alert prediction task as well. Lastly, the DL system presently considers only those SourceIPs which generate more that 100 alerts in a month. More efforts are required in order to tweak the system to include less frequent sources.

7. Conclusion

This paper presented two distinct approaches for intrusion alert prediction. The first approach was a shallow learning approach which required feature engineering on the alert data and assigned a probabilistic score to each attacking source. This reputation score can be used as an indication of the future behavior of a malicious entity. Thereafter, another solution based on Convolutional LSTM DL network was presented. The DL model was able to predict future alert sequences originating from attacking sources with acceptable accuracy. For both the proposed solutions, network intrusion alerts provided by an alert sharing system called Warden were used for the training and evaluation purposes. Future research endeavors shall focus on combining the two approaches for a more holistic prediction performance.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 Research and Innovation Program, PROTECTIVE, under Grant Agreement No. 700071.

References

[1] Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A., 2011. Sequential deep learning for human action recognition, in: International Workshop on Human Behavior Understanding, Springer. pp. 29–39.
[2] Bartos, V., Zadnik, M., Habib, S.M., Vasilomanolakis, E., 2019. Network entity characterization and attack prediction. Future Generation Computer Systems 97, 674–686.
[3] Connor, J.T., Martin, R.D., Atlas, L.E., 1994. Recurrent neural networks and robust time series prediction. IEEE transactions on neural networks 5, 240–254.
[4] Du, M., Li, F., Zheng, G., Srikumar, V., 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM. pp. 1285–1298.
[5] Graves, A., Mohamed, A.R., Hinton, G., 2013. Speech recognition with deep recurrent neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on, IEEE. pp. 6645–6649.
[6] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.
[7] Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., Atkinson, R., 2017. Shallow and deep networks intrusion detection system: A taxonomy and survey. arXiv preprint arXiv:1701.02145 .
[8] Holgado, P., Villagra, V.A., Vazquez, L., 2018. Real-time multistep attack prediction based on hidden markov models. IEEE Transactions on Dependable and Secure Computing , 1–1.
[9] Jia, R., Liang, P., 2016. Data recombination for neural semantic parsing. arXiv preprint arXiv:1606.03622 .
[10] Malhotra, P., Vig, L., Shroff, G., Agarwal, P., 2015. Long short term memory networks for anomaly detection in time series, in: Proceedings, Presses universitaires de Louvain. p. 89.
[11] Moura, G., Sadre, R., Pras, A., 2014. Bad neighborhoods on the internet. IEEE Communications Magazine 52, 132–139.
[12] Okutan, A., Werner, G., McConky, K., Yang, S.J., 2017. Poster: Cyber attack prediction of threats from unconventional resources (capture), in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, New York, NY, USA. pp. 2563–2565.
[13] Ramaki, A.A., Khosravi-Farmad, M., Bafghi, A.G., 2015. Real time alert correlation and prediction using bayesian networks, in: 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), pp. 98–103.
[14] Schmidhuber, J., Wierstra, D., Gomez, F.J., 2005. Evolino: Hybrid neuroevolution/optimal linear search for sequence prediction, in: Proceedings of the 19th International Joint Conferenceon Artificial Intelligence (IJCAI).
[15] Shen, Y., Mariconti, E., Vervier, P.A., Stringhini, G., 2018. Tiresias: Predicting security events through deep learning, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ACM. pp. 592–605.
[16] Soldo, F., Le, A., Markopoulou, A., 2011. Blacklisting recommendation system: Using spatio-temporal patterns to predict future attacks. IEEE Journal on Selected Areas in Communications 29, 1423–1437.
[17] Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M., Fischer, M., 2015. Taxonomy and survey of collaborative intrusion detection. ACM Computing Surveys (CSUR) 47, 55.
[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: Advances in neural information processing systems, pp. 5998–6008.
[19] Wang, J., Yi, Y., Zhang, H., Cao, N., 2018. Network attack prediction method based on threat intelligence, in: Sun, X., Pan, Z., Bertino, E. (Eds.), Cloud Computing and Security, Springer International Publishing, Cham. pp. 151–160.
[20] Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: Advances in neural information processing systems, pp. 802–810.
[21] Zhang, J., Porras, P., Ullrich, J., 2008. Highly predictive blacklisting, in: USENIX Security Symposium, USENIX Association, Berkeley, CA, USA. pp. 107–122.