Host Based Intrusion Detection System with Combined CNN/RNN Model

Ashima Chawla(✉), Brian Lee, Sheila Fallon, and Paul Jacob

Athlone Institute of Technology, Athlone, Ireland
a.chawla@research.ait.ie, {blee,sheilafallon,pjacob}@ait.ie

Abstract. Cyber security has become one of the most challenging aspects of modern world digital technology and it has become imperative to minimize and possibly avoid the impact of cybercrimes. Host based intrusion detection systems help to protect systems from various kinds of malicious cyber attacks. One approach is to determine normal behaviour of a system based on sequences of system calls made by processes in the system [1]. This paper describes a computational efficient anomaly based intrusion detection system based on Recurrent Neural Networks. Using Gated Recurrent Units rather than the normal LSTM networks it is possible to obtain a set of comparable results with reduced training times. The incorporation of stacked CNNs with GRUs leads to improved anomaly IDS. Intrusion Detection is based on determining the probability of a particular call sequence occurring from a language model trained on normal call sequences from the ADFA Data set of system call traces [2]. Sequences with a low probability of occurring are classified as an anomaly.

Keywords: Host based intrusion detection systems (HIDS) · Gated Recurrent Unit (GRU) · System calls · Recurrent Neural Network (RNN) · Convolutional Neural Network (CNN)

1 Introduction

In recent years with the advancement of technology, cyber security has become a major concern due to the high level of attacks on organization networks and systems. In such scenarios, Intrusion Detection Systems (IDS) are a crucial requirement to safeguard an organization's electronic assets. There are two types of intrusion detection systems commonly known as Host based Intrusion Detection systems (HIDS) and Network based Intrusion Detection systems (NIDS).

Network based intrusion detection systems are used to monitor and analyze network traffic to protect a system from network-based threats. Network based IDS aims at collecting information from the packet itself and looks at the contents of individual packets with the aim to detect the malicious activity in network traffic. Host based intrusion detection systems are a network security technology

C. Alzate et al. (Eds.): ECML PKDD 2018 Workshops, LNAI 11329, pp. 149–158, 2019.
https://doi.org/10.1007/978-3-030-13453-2_12

originally built for detecting vulnerability exploits against a target application or computer system. A HIDS aims to collect information about events or system calls/logs on a particular system.

The two main types of HIDS are signature-based and anomaly based. The signature based approach operates in much the same way as a virus scanner, by searching for identities or signatures of known intrusion events, while the anomaly based approach establishes a baseline of normal patterns. Anomaly based IDS allows the detection of unseen attacks, though resulting in higher false alarm rates but when paired with signature detection, can result in a powerful defense.

System calls or *kernel* calls provide an essential interface between a process and the operating system. Forrest was the first to suggest that sequences of system calls could be used to capture normal behaviour in a computer system [1]. In this context, Australian Defence Force Academy Linux Dataset (ADFA-LD), a recently released system call dataset consists of 833 normal training sequences, 746 attack, 4372 validation sequences and has been used for evaluating a system call based HIDS. The system call traces consists of call sequences of integers. Due to the diverse and dynamic nature of system call patterns, it becomes difficult to separate the normal and abnormal behaviours.

Over the past few years, sequence to sequence learning has achieved remarkable success in the field of machine learning tasks such as speech recognition, language models [3,4] and text summarization [5–7] amongst others. Convolutional Neural Networks (CNNs) were shown to perform well on certain sequence processing problems at a considerably cheaper computational cost than Recurrent Neural Networks (RNNs) and the combined architecture of CNN-RNN as described in [8] was able to achieve high accuracy for sentiment analysis in short text.

Motivated by these applications in the domain of Deep Neural Networks, we propose an architecture with two significant contributions. Firstly, to model sequence to sequence learning which is a combination of a multilayer CNN with an RNN made up of Gated Recurrent Units (GRUs) where local features in the input sequences are extracted by the CNN layer and used as an input to the GRU layer. The output from the GRU layer is processed with a fully connected softmax layer that outputs a probability distribution over system call integers, resulting in an architecture similar to [9]. Secondly, with reduced training times, we were able to effectively replace LSTM with GRU and obtain a set of comparable results.

2 Related Work

A smart Intrusion detection system can only be implemented if we have an effective dataset. Several researchers have adopted various algorithms to achieve the state of art in detecting anomalous data. This section briefly discusses the various algorithms and frameworks designed so far developed to detect intrusions.

Early in 1990 and 2000, Knowledge Discovery in Databases (KDD98) and UNM (2004) datasets were released for evaluating intrusion detection systems.

Creech [2] claimed that the testing of new intrusion detection system algorithms against these datasets was no longer relevant as the datasets were not representative of modern attacks. In 2012, the ADFA dataset was made publicly available to aid the researchers to represent true performance against contemporary modern attacks. The ADFA-LD data set [2] was published as a proposed replacement for the widely used KDD98 dataset and was seen to contain low foot print attacks [17] so that abnormal data become quite homogeneous and difficult to separate. The ADFA-LD data had been collected using the Linux audit daemon.

A **Window based** approach as adopted by Forrest et al. [1] extracts a fixed size windows system call sequence as a trace generally represented as a feature vector, which proved to be quite ineffective against handling sufficiently long traces where anomalous call sequences are quite dispersed. Kosoresow et al. [10] proposed another window frames based algorithm to determine the locality of anomalies within a trace by partitioning each trace into a number of small and fixed length sections called locality frames, but which often results in a time consuming learning procedure.

Later, a **Frequency based** approach as adopted by Xie et al. [16] attempted to implement an efficient *kNN* based HIDS using the concept of frequency of system call traces, which achieved a Detection rate of around 60% with an approximate 20% False Alarm rates.

In [11], the authors employed discontiguous system call patterns and claimed that original semantic feature based ELM (Extreme Learning Machine) turned out to be superior to all other algorithms and obtained Detection rate of 90% with 15% False Alarm rate but with the major drawback of a high computational time. Marteau [20] introduced the concept of an efficient algorithm (SC4ID), also known as Sequence Covering For Intrusion Detection system and achieved AUC of 0.842 using the kernel based family approach. However, the above stated kernel based methods proved inadequate to capture inter-word (system calls) relationships and sentence (system-call sequences) structure.

Recently, a **Sequential Language** model approach calculates the probability distribution over the sequence of words and has gained remarkable performance in terms of capturing inter word relationships. One of the recent approaches by Kim et al. [19] proposed an intrusion detection system using Long Short Term Memory which captured the semantic meaning of each call and its relation to other system calls. We apply a similar concept to explore what factors our models attend over when predicting anomaly scores with reduced training times using stacked CNN over GRU.

3 Methodology

3.1 Recurrent Neural Networks

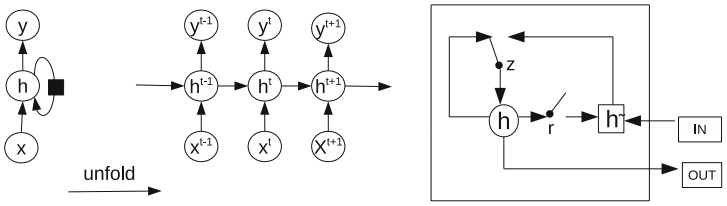
A feed-forward neural network has an input layer, a number of hidden layers and an output layer. The output for a node in the network is obtained by applying a weight matrix to the node's inputs and applying an activation function to the

result. The network is trained using an algorithm such as backpropagation. This involves calculating gradients for each weight in the neural network and using these to adjust each weight so that the network produces the output required.

Recurrent Neural Networks (RNNs) are a form of network with backward connections, where output from a layer in the network is fed back into either that layer or a previous layer in the network [12]. RNNs maintain state, in that values calculated at a previous timestep are used in the current timestep. This state is used as a form of short term memory in the network and RNNs are typically used as a model for time series and sequential data where values at previous time steps can affect the current calculation.

As shown in Fig. 1(a), RNNs can be unfolded to become a regular neural network. In this diagram a single node represents a complete layer in the RNN. Backpropagation applied to this unfolded network is known as Backpropagation Through Time and can be used to train the RNN. While RNN can be trained to capture short term dependencies between time steps, it has proved difficult to train RNNs to capture long term dependencies due to the so called "vanishing gradient" problem. To overcome this, special types of RNNS have been designed, in particular Long Short Term Memory networks (LSTM) and Gated Recurrent Units (GRU).

LSTM networks have an LSTM cell that stores state over time [13]. Input gates, output gates and forget gates provide access to these cells in such a way that values can be stored in the cell for either short or long periods of time, and removed when no longer needed. LSTMs have been shown to overcome the vanishing gradient problem of ordinary RNNs. As shown in Fig. 1(b) GRUs have an update and reset gate and have fewer parameters than LSTMs and are faster to train [14].



(a) Unfolding an RNN (b) GRU Diagram [14]

Fig. 1. RNN model architecture

3.2 1D Convolutional Neural Networks

Convolutional Neural networks are a type of network primarily used in image processing but with other applications as well. In the case of 2D data, convolution is effected by a 2D filter sliding over the image and applying some function to the

covered part of the image to produce the output. By using suitable functions, patterns in the image can be detected, for example, taking the difference between pixels can be used to detect edges.

In the case of 1D data, filters slide over sequences extracting a feature map for local sub-sequences in the data. They create representations for fixed size contexts and the effective context size can easily be made larger by stacking several CNN layers on top of each other. This allows to precisely control the maximum length of dependencies to be modeled. As convolutions are a common operation in computer graphics with direct hardware support on GPUs, CNNs are a more efficient way of extracting local patterns from sequences than RNNs. Note that following [18], pooling is not applied after the convolution operation. The output from the stacked CNN layers are passed to the RNN which can be used to capture long-range dependencies.

3.3 Sequence Anomaly Detection Using Language Modeling

In the ADFA-LD data set, system calls are represented as integers in the range 1 to 340. Following [19] let $x = x_1, x_2, \ldots, x_l$, where x_i is an integer. A language model for system call sequences specifies a probability distribution for the next call in a sequence given the sequence of previous system calls. The Neural Network is trained to produce this probability distribution using a training set of known normal sequences, that is, the network learns a language model of normal sequences.

We can estimate the probability of a sequence occurring using these probability distributions. Note that $p(x_i|x_{1:i-1})$ is the probability of the integer x_i occurring after the sequence $x_{1:i-1}$.

$$p(x) = \prod_{i=1}^{l} p(x_i|x_{1:i-1}) \tag{1}$$

In practice the negative log of the value p(x) defined in Eq. (1) is used resulting in high values for unlikely sequences and low values for likely sequences. Anomaly detection for sequences can be carried out by imposing a threshold for this negative log likelihood (L) and predicting an anomaly for sequences with an L value above this threshold.

4 Experimental Setup and Results

In this section, we outline five models of different combinations of GRU, LSTM and CNN, presenting ROC curves for each and compare with other results. An overview of model architecture is presented in Sect. 4.1. Section 4.2 outlines the model definitions providing various hyperparameters and Sect. 4.3 evaluates the experimental results.

The ADFA Intrusion detection dataset [2] consists of 833 normal training sequences as well as 4372 normal validation and 746 attack sequences for

testing. The specification of the computational machine includes Intel core i7-8700@3.20 GHz processor, 16 GB of RAM and NVIDIA GeForce GTX1070 GPU running 64 bit Windows 10 operating system and the NVIDIA CUDA® Deep Neural Network library (cuDNN). The **Keras** python library [15] was used running on top of a source build of Tensorflow 1.7.1 with CUDA support. For the purposes of evaluation, Detection Rate (DR) and False Alarm Rates (FAR) were defined as:

$$DR = TP/(TP + FN) \tag{2}$$

$$FAR = FP/(FP + TN) \tag{3}$$

4.1 Model Architecture

The Keras model we built consists of a number of layers as described below in Fig. 2.

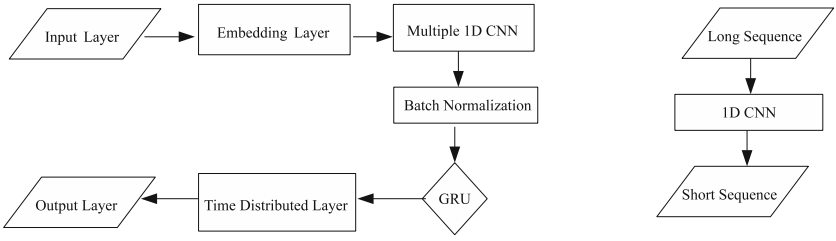


Fig. 2. HIDS model architecture

The Keras embedding layer performs word embedding and transforms one-hot encoding of integers in the call sequence, which vary from 1 to 340, into a dense vector of size 32. Embedding layer weights are learned during training, that is a pre-trained embedding is not used. The 1D CNN layer in Keras (Conv1D layer) processes input batches independently and as they aren't sensitive to the order of the time steps, can be executed in parallel.

Thus, 1D convnets nets are used as a pre-processing step to make the sequence smaller resulting in a faster training. In practice, the CNN layers extract higher level local features, which are then passed on to the GRU as input. The Keras Batch Normalization layer helps with gradient propagation and significantly increases the training speed. The GRU layer, with a Keras parameter "return sequences" set to true returns the hidden state output for each input time step and is necessary when passing data to the TimeDistributed Layer.

The TimeDistributed Layer is an example of a Keras Wrapper Layer. It applies a same Dense (fully-connected) operation to every timestep of a 3D input and allows us to gather the output at each timestep, effectively supporting sequence to sequence learning. The output layer is a Keras Dense layer, essentially a regular densely connected neural network layer. It is used with a softmax activation function in order to predict a probability distribution for the next integer in the call sequence.

4.2 Model Definitions

Accordingly, we built five independent models: (1) one layer with 200 GRU units (2) one layer with 200 LSTM units (3) Six layered 1D CNN with 200 GRU units (4) Seven layered 1D CNN with 500 GRU units (5) Eight layered 1D CNN with 600 GRU units. Each model was trained with 833 normal sequences, which were processed in variant length mini batches, where each sequence in a mini batch was padded to the length of the longest system call in the mini batch. We used Adam optimizers with a learning rate of 0.0001, a softmax activation function in Time Distributed layer and relu activation function at the CNN layer with drop out probability of 0.7 before the softmax layer.

4.3 Experimental Results

Equation (1) was used to calculate an overall probability for the sequence where Fig. 3 shows the ROC curves for the above outlined models.

The model with CNN+GRU 600 units gave the best value (0.81) for the Area Under the ROC curve (AUC). CNN+GRU 200 and 500 units were only marginally behind resulting in an AUC value of 0.80. The model produces 100% True Detection Rate with a False Alarm Rate of 60%.

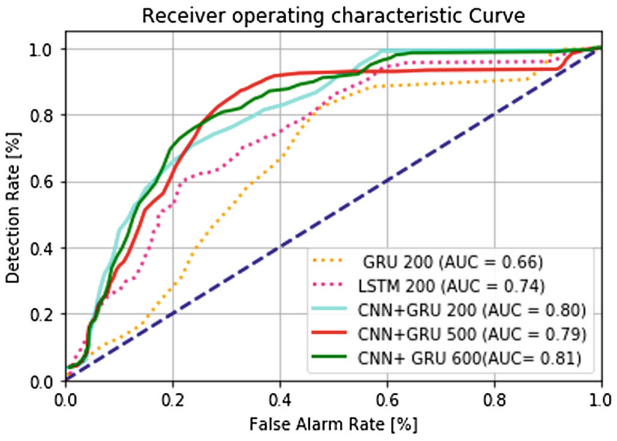


Fig. 3. ROC curve comparing different models of ADFA dataset

5 Analysis

We have shown that the CNN-GRU language model implementation has substantially reduced the training time when compared to an LSTM model.

Secondly, we were able to achieve better accuracy by stacking multiple CNN layers before the GRU layer. The time taken for stacked CNN/GRU is approximately 10 times faster than LSTM due to faster convergence in training. While

the CNN-GRU model converged after 10 training epochs, giving an AUC of 0.80, the LSTM model needed 100 epoch to converge resulting in an AUC of 0.74 with 100 epochs (Table 1).

Table 1. Model analysis

Model	RNN units	Training time (sec)	Testing time (sec)	AUC
GRU	200	376	444	0.66
LSTM	200	4444	541	0.74
CNN+GRU	200	390	441	0.80
CNN+GRU	500	402	493	0.79
CNN+GRU	600	413	533	0.81

Additionally, in LSTM based sequence modeling paper [19], the authors was able to achieve the True Detection rate of 100% and false alarm rate of 50–60%, while training the normal 833 sequences using LSTM method, comparatively we were able to achieve the results with 100% True Detection Rate and the false alarm rate of 60% using combined CNN/GRU method.

For future work we intend to determine if increasing the number of training samples will improve anomaly detection. With improved training execution time this would now be feasible. Secondly we intend to implement various other algorithms such as a kNN based model, and an Encoder-Decoder model based on sequence reconstruction error. Finally, as demonstrated in [19], an ensemble method will most likely give the best results and we plan to build and evaluate such a model.

6 Conclusion

In this paper we propose a CNN-GRU language model for the recently released ADFA-LD intrusion detection data set. As outlined in [18], the CNN layers can capture local correlations of structures in the sequences and can execute in parallel improving performance while the RNN (GRU) layer can then learn sequential correlations from these higher level features.

The model is trained on normal call sequences and predicts a probability distribution for the next integer in a call sequence. This in turn is used to predict a probability for the entire sequence and a threshold for classification is chosen from the range of negative log likelihood values. We have maintained near state of art performance for neural network models with a substantial reduction in training times compared to LSTM models. We have been unable to match the performance of ensemble models [19] but that is to be expected. Our model should be a useful part of an overall ensemble model, possibly combined with a KNN based model and an encoder-decoder model.

Acknowledgments. This paper has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No. 700071 for the PROTECTIVE project.

References

1. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for Unix processes. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA, pp. 120–128 (1996)
2. Creech, G., Hu, J.: Generation of a new IDS test dataset: time to retire the KDD collection. In: IEEE Wireless Communications and Networking Conference (WCNC) (2013)
3. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS 2014 Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, pp. 3104–3112, December 2014
4. Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition. In: NIPS 2014 Deep Learning Workshop (2014)
5. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of EMNLP (2015)
6. Nallapati, R., Zhou, B., dos Santos, C.N., Gulcehre, C., Xiang, B.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: The SIGNLL Conference on Computational Natural Language Learning (CoNLL) (2016)
7. Shen, Y., Huang, P.-S., Gao, J., Chen, W.: ReasoNet: learning to stop reading in machine comprehension. Microsoft Research: Neural and Evolutionary Computing (cs.NE), 17 Sept 2016
8. Wang, X., Jiang, W., Luo, Z.: Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–17 December 2016, pp. 2428–2437 (2016)
9. Sainath, T.N., Vinyals, O., Senior, A., Sak, H.: Convolutional, long short-term memory, fully connected deep neural networks. In: Google, Acoustics, Speech and Signal Processing (ICASSP), pp. 4580–4584. IEEE (2015)
10. Kosoresow, A.P., Hofmeyr, S.A.: Intrusion detection via system call traces. IEEE Softw. **14**(5), 35–42 (1997)
11. Creech, G., Hu, J.: A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. IEEE Trans. Comput. **63**, 807–819 (2014)
12. Graves, A.: Supervised Sequence Labeling with Recurrent Neural Networks. SCI. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24797-2
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. Presented at the Deep Learning Workshop at NIPS, arXiv:1412.3555 (2014)
15. Keras Home Page. https://keras.io/. Accessed 7 July 2018
16. Xie, M., Hu, J.: Evaluating host-based anomaly detection systems: a preliminary analysis of ADFA-LD. In: 6th International Congress on Image and Signal Processing (CISP), Hangzhou, China (2013)

17. Haider, W., Hu, J., Xie, M.: Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems. In: IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, New Zealand (2015)
18. Zhou, C., Sun, C., Liu, Z., Lau, F.C.M.: A C-LSTM neural network for text classification. arXiv:1511.08630 (2015)
19. Kim, G., Yi, H., Lee, J., Paek, Y., Yoon, S.: LSTM-Based System-Call Language Modeling and Robust Ensemble Method for Designing Host-Based Intrusion Detection Systems. eprint arXiv:1611.01726 (2016)
20. Marteau, P.-F.: Sequence Covering for Efficient Host-Based Intrusion Detection (2017)